Martin Klotz

Studie zu Steuerungssystemen der Firma „Bernecker und Rainer Industrie-Automation GesmbH“

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Maschinenbau / Mechatronik

Mittweida, 2011

Erstprüfer: Prof. Dr.-Ing. Dietmar Römer
Hochschule Mittweida

Zweitprüfer: Dipl. Ing. Albert Hohenwarter
SANDOZ GesmbH

Vorgelegte Arbeit wurde verteidigt am:

Bibliographische Beschreibung:

Klotz, Martin:

Studie zu Steuerungssystemen der Firma „Bernecker u. Rainer Industrie Elektronik GesmbH“. - 2011. – 60 S. Mittweida, Hochschule Mittweida, Fakultät Maschinenbau, Diplomarbeit, 2011

Referat:

Ziel der Diplomarbeit ist es, eine Studie zu den von der Firma „Bernecker u. Rainer Industrie Elektronik GesmbH“ angebotenen Steuerungssystemen zu erstellen.

Diese soll die Leistungsfähigkeit der Steuerungssysteme aufzeigen und dadurch Aufschluss über die Einsatzmöglichkeiten der Komponenten geben. Es sollen die Eigenschaften der Zentraleinheiten, der Ein- und Ausgangsbaugruppen und der kompatiblen Bussysteme näher gebracht werden.

Weiters soll die Software zur Programmierung der Systeme, „Automation Studio“ von B+R, untersucht werden. Mittels Programmbeispielen inklusive entsprechender Visualisierung soll die Programmierung und visuelle Darstellung von SPS-Programmen in Automation Studio getestet werden. Dadurch soll die Bedienbarkeit und Verständlichkeit von Automation Studio bewertet werden.

Um den Einsatz des Steuerungssystems X20 von Bernecker und Rainer in industriellen Anwendungen aufzuzeigen, soll im Zuge dieser Arbeit auch ein Programmbeispiel ausgearbeitet werden, in dem verschiedene Feldkomponenten an die I/O-Baugruppen der Hardware angebunden, im Programm verarbeitet und über die integrierte Visualisierung bedienbar und auslesbar gemacht werden.

Außerdem sollen in dieser Arbeit die Funktion und die Leistungsdaten des industriellen Ethernet-Standards „Ethernet Powerlink“ beschrieben werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis (alphabetisch sortiert)

AS (SFC)	Ablaufsprache (Sequential Function Chart)
AWL (IL)	Anweisungsliste (Instruction List)
B+R	Bernecker und Rainer Industrie Elektronik GesmbH
CAN	Controller Area Network
CFC	Continous Function Chart
CiA	CAN in Automation
CIP	Common Industrial Protocol
CN	Controlled Node
CPU	Central Processing Unit
CSMA/CR	Carrier Sense Multiple Acces / Collision Resolution
EMV	Elektromagnetische Verträglichkeit
EN	Europäische Norm
EPSG	Ethernet Powerlink Standardization Group
FBS (FBD)	Funktionsbausteinsprache (Function Block Diagram)
I/O	Input / Output (Eingang / Ausgang)
I/P	Elektrischer Strom / pneumatischer Druck
IEC	International Electrotechnical Comission
IT	Informationstechnologie
KOP (LD)	Kontaktplan (Ladder Diagram)
MN	Managing Node
ODVA	Open DeviceNet Vendor Association
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OSI	Open System Interconnection
PC	Personal Computer
PDO	Prozessdatenobjekt
PG	Programmiergerät
PReq	Poll Request
PRes	Poll Response
PVI	Process Visualization Interface
RPS	Rights Protection System
SCADA	Supervisory Control and Data Acquisition
SDO	Systemdatenobjekt
SoC	Start of Cycle
SPS	Speicherprogrammierbare Steuerung
ST (ST)	Strukturierter Text (Structured Text)
TCP/IP	Transmission Control Protocol / Internet Protocol
USB	Universal Serial Bus
VNC	Virtual Network Computing

Tabellenverzeichnis

1. Das Unternehmen Bernecker und Rainer GesmbH[1]

Die Bernecker und Rainer Industrie Elektronik GesmbH wurde im Jahr 1979 von Erwin Bernecker und Josef Rainer gegründet und zählt heute mit Standorten in über 70 Ländern und etwa 2000 Mitarbeitern zu den größten Privatunternehmen im Bereich von integrierten Automatisierungslösungen.

Zur Produktpalette von B+R zählen unter anderem:

- Industrie PC's
- Visualisierungs- und Bedientechnik
- **Steuerungssysteme**
- Antriebstechnik
- Netzwerk- und Feldbusmodule
- Software
- Stromversorgungen
- …

Abbildung 1: Steuerungskomponenten B+R

Unter dem Leitspruch „Perfection in Automation" avancierte die Firma Bernecker und Rainer in den letzten 30 Jahren von einem kleinen Unternehmen in der Automatisierungsbranche zu einem der renommiertesten Anbieter dieses Geschäftsbereiches. Der Umsatz betrug im Jahr 2010 bereits mehr als 300 Mio. Euro.

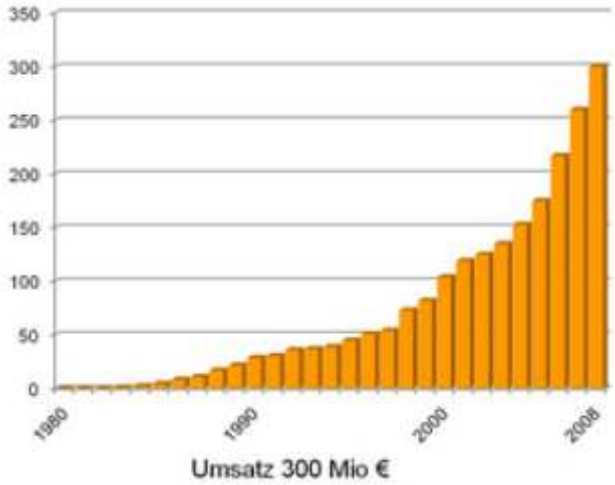

Abbildung 2: Umsatzdiagramm B+R

1 URL http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/company_DEU_HTML.htm [01.12.2010]

2. Steuerungssysteme B+R

Es gibt mittlerweile 3 Generationen der Steuerungssysteme von B+R, welche in den folgenden Punkten beschrieben werden. Da im Zuge dieser Diplomarbeit das X20-System zur Anwendung kommt, wird dieses System detaillierter als die Vorgängersysteme behandelt.

2.1. System 2003

2.1.1. Eigenschaften[2]

Hardware:

- Hardwaremodularität
- Vernetzbarkeit
- Kommunikationsschnittstellen zu HMI
- RPS- und Industrierechnerfunktionalität
- EMV-Verträglichkeit nach EN61131-2
- Abgesichertes I/O-Busprotokoll
- Dezentrale I/O-Punkte
- Bit- oder Wortverarbeitung in nur einem Zyklus
- Alle Klemmpunkte für 2- oder 3-Leiteranschluss direct am Modul
- Keine Zwischenklemmen erforderlich

Software:

- Multitasking-Betriebssystem für RPS- und Industrierechneranwendungen
- Leistungsfähige RPS Programmiersprachen
- Hochsprachenprogrammierung
- Exakte Kontrolle über das Zeitverhalten der Steuerungssysteme
- Einfach zu bedienende Programmiersoftware mit fensterorientierter Oberfläche
- Projektverwaltung im Programmiergerät

2 "Anwenderhandbuch System 2003", Bernecker und Rainer GesmbH, 2000;

2.1.2. Programmierung[3]

Die Programmierung der Komponenten des Systems 2003 erfolgt entweder über das Programmiergerät PG 2000 oder mit der B+R Programmiersoftware “Automation Studio”.

Die folgende Grafik zeigt die mit dem System 2003 kompatiblen Programmiersprachen.

PG2000	**Automation Studio™**
Anweisungsliste (AWL)	Automation Basic (vormals PL2000)
Kontaktplan (KOP)	ANSI C
Hochsprache PL2000 (strukturierter Text)	IEC 1131 Kontaktplan (KOP)
	IEC 1131 Ablaufsprache (AS)
	IEC 1131 Strukturierter Text (ST)
	IEC 1131 Anweisungsliste (AWL)

Tabelle 1: Programmiersprachen System 2003

2.2. System 2005

2.2.1. Eigenschaften[4]

Hardware:

- Hardwaremodularität
- Vernetzbarkeit
- Kommunikationsschnittstellen zu HMI
- RPS- und Industrierechnerfunktionalität
- EMV-Verträglichkeit nach EN61131-2
- Abgesichertes I/O-Busprotokoll
- Dezentrale I/O-Punkte
- Kombinierter I/O- und Systembus
- Bit- oder Wortverarbeitung in nur einem Zyklus
- Leistungssteigerung durch mehrere I/O-Bussysteme

3 “Anwenderhandbuch System 2003“, Bernecker und Rainer GesmbH, 2000;
4 “Anwenderhandbuch System 2005“, Bernecker und Rainer GesmbH, 2004;

Software:

- Multitasking-Betriebssystem für RPS- und Industrierechneranwendungen
- Leistungsfähige RPS Programmiersprachen
- Hochsprachenprogrammierung
- Exakte Kontrolle über das Zeitverhalten der Steuerungssysteme
- Einfach zu bedienende Programmiersoftware mit „Windows“ Oberfläche
- Projektverwaltung im Programmiergerät

2.2.2. Programmierung[5]

Die Programmierung der Komponenten des Systems 2005 erfolgt über die Programmiersoftware von B+R. (Automation Studio)

Das System 2005 ist kompatibel mit folgenden Programmiersprachen:

- Automation Basic
- ANSI C
- Kontaktplan
- Ablaufsprache
- Strukturierter Text
- Anweisungsliste

2.3. System X20[6]

Das X20 Steuerungssystem ist die aktuellste Serie der B+R Steuerungssysteme. Das System wurde auf der SPS/IPC/Drives 04 vorgestellt und ging im Frühjahr 2005 in Serienproduktion. In den folgenden Punkten werden die grundlegenden Funktionen und Eigenschaften des X20-Systems beschrieben.

5 “Anwenderhandbuch System 2005“, Bernecker und Rainer GesmbH, 2004;
6 “Anwenderhandbuch System X20“, Bernecker und Rainer GesmbH, 2009;

2.3.1. Eigenschaften

2.3.1.1. Geteilte Hardware-Baugruppen[7]

Die Baugruppen des X20-Systems sind in 3 Module aufgeteilt, was in vielerlei Hinsicht Vorteile bietet:

- Busmodule: Das Busmodul ist der Grundbaustein aller Baugruppen. Es bindet das aufgesetzte Elektronikmodul an die Backplane an und versorgt es mit der Betriebsspannung und der I/O-Spannung. Ein ausgeklügeltes System zur Hutschienenmontage macht eine schnelle Installation im Schaltschrank möglich.

- Elektronikmodule: Die Elektronikmodule der einzelnen Baugruppen sind sehr einfach während des Betriebes auszutauschen, was eine Erweiterung des Systems während des Betriebes erlaubt. Das System erkennt den Ausbaugrad der Baugruppe automatisch und stellt somit sofort nach dem Stecken des Moduls die notwendige Funktionalität zur Verfügung.

- Klemmenmodule: Die Klemmenmodule des X20 Systems sind jene Baugruppe, die die Buchsen zum Anschluss der Feldgeräte zur Verfügung stellen, und somit die Schnittstelle des Steuerungssystems zur Sensorik und Aktorik der Anlage. Die Klemmenmodule sind abnehmbar ausgeführt und erlauben so eine Vorverdrahtung von kompletten Schaltschränken. Dies ist ein maßgeblicher Vorteil des Systems in Bezug auf Serienmaschinenbau.

Abbildung 3: 3x1 System von B+R

7 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

2.3.1.2. Dezentrale Backplane[8]

Die Kernidee des X20 Systems steckt in der dezentralen Backplane. Dezentral bedeutet, dass die Verbindungskabel zwischen mehreren X20 Nestern selbst, quasi die Backplane sind und nicht nur die Verbindung zwischen den einzelnen Backplanes darstellen. Das Verbindungskabel (die Backplane) nennt sich X2X-Link und besteht zur Erhöhung der Störsicherheit aus verdrillten Kupferkabeln. Durch diese dezentrale Backplane lässt sich eine Kommunikation zwischen allen in einem X2X-Link-Netzwerk enthaltenen Komponenten nahezu ohne Performance-Verlust bewerkstelligen. Eine weitere Funktion des X20-Systems ist, dass mittels nicht belegter Busmodule nachträglich Elektronikmodule eingefügt werden können, ohne eine Änderung an der Softwareadressierung durchführen zu müssen. Eine Verbindung zwischen 2 X20-Systemen kann bis zu 100 Meter lang sein.

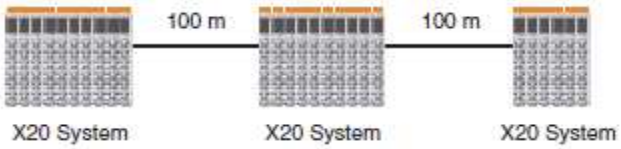

Abbildung 4: Verbindung von X20 Systemen

2.3.1.3. CPU's und deren Leistungsdaten[9]

Abbildung 5: X20 Zentraleinheit

Die derzeit von B+R angebotenen dem X20 System angehörenden Zentraleinheiten decken ein sehr breites Leistungsspektrum ab. Es kann von kleinen Standardsteuerungen bis hin zu Automatisierungsprojekten mit großem Umfang eigentlich alles realisiert werden. Bei fast allen CPU's der Baureihe gehören RS232, Ethernet und USB-Schnittstellen bereits zur Standardausstattung, womit ohne jegliche Zusatzkosten die Netzwerkfähigkeit und die Möglichkeit zum Anschluss von USB-Geräten geboten sind. Der USB-Anschluss funktioniert jedoch nur in Zusammenarbeit mit original von B+R angebotenen Komponenten. Falls die standardmäßig enthaltenen Schnittstellen nicht ausreichend sind, enthalten die Zentraleinheiten bis zu 3 Steckplätze für verschiedenste Kommunikationsmodule. An die CPU können **bis zu 250 I/O-Module** direkt angebunden werden, was in etwa 3000 Mess- und Steuerkanälen entspricht.

8 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;
9 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

2.3.1.4. Leistungsdaten der X20-Zentraleinheiten[10]

In der folgenden Tabelle werden die Leistungsdaten der Zentraleinheiten der X20-Serie aufgezeigt.

Kurzbeschreibung	**CP1483**	**CP1484**	**CP3484**	**CP1485**	**CP3485**	**CP1486**	**CP3486**
Prozessor	X86 100	Celeron 266		Celeron 400		Celeron 650	
Schnittstellen	1 x RS232, 1 x Ethernet, 1 x Ethernet Powerlink V1/V2, 2 x USB, 1 x X2X Link						
Schnellste Tasklassen Zykluszeit	1 ms	800 µs		400 µs		200 µs	
Typische Befehls-Zykluszeit	0,076 µs	0,022 µs		0,015 µs		0,01µs	
Daten- und Programmcode L1 Cache	16 kByte	2 x 16 kByte					
L2 Cache	---			256 kByte			
SDRAM (Arbeitsspeicher)	32 MByte	32 MByte		64 MByte			
SRAM (User RAM)	128 kByte	1 MByte		1 MByte			
Versorgungsspannung	24 VDC						
Betriebstemperatur	0 ℃ bis +55 ℃ (hängt jedoch von Einbaulage ab)						
Luftfeuchtigkeit	5 bis 95%, nicht kondensierend						

Tabelle 2: Leistungsdaten X20 CPU's

10 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

2.3.1.5. Diagnosefunktionen[11]

Ein wichtiges Merkmal von X20 Steuerungssystemen in der modernen Automatisierungstechnik sind die zur Verfügung stehenden Diagnosefunktionen. Folgende Diagnosefunktionen sind bei den Systemen der Serie X20 von B+R verfügbar:

- Direkt an den Modulen befinden sich LED-Anzeigen für Busstatus, I/O-Status und Kanalzustände der Baugruppe. Dies ermöglicht vor Ort bereits eine erste Diagnose ohne Programmiergerät und Werkzeug.
- Über „Automation Studio“ kann das zyklische Prozessdatenabbild geladen werden. In diesem sind alle relevanten Daten über I/O-Zustände, Statusmeldungen der Buskommunikation und die Zustände bzw. Statusmeldungen der angekoppelten Baugruppen enthalten.
- Auch azyklisch, also asynchron zum normalen Datenverkehr können ohne einen Performanceverlust ausführliche Daten zur Diagnose eines Moduls angefordert werden. Dabei bleiben die Zykluszeiten unverändert, das heißt es entsteht keine Einschränkung für die Kommunikationsfunktionen des Systems.

Abbildung 6: Diagnose und Zustandsanzeigen X20

Diese Diagnosefunktionen erlauben und ermöglichen im Fehlerfall eine effiziente und schnelle Lokalisierung des Problems und stellen damit ein unentbehrliches Werkzeug für Programmierer und vor allem Instandhaltung dar.

11 “Anwenderhandbuch System X20“, Bernecker und Rainer GesmbH, 2009;

2.4. Kompatibilität mit CoDeSys

2.4.1. CoDeSys[12]

CoDeSys steht für Controller Development System. Es handelt sich um eine Programmiersoftware für Steuerungen, Embedded und PC-basierte Steuerungsgeräte. Die Vollversion steht nach Registrierung auf der Homepage www.3s-software.com gratis zum Download zur Verfügung. Entwickelt und betreut wird die Software von der Firma 3S-Smart Software, deren Firmensitz in Deutschland, 87439 Kempten, Memminger Straße 151 liegt. Das Unternehmen wurde im Jahre 1994 von Dieter Hess und Manfred Werner gegründet und beschäftigt mittlerweile bei einem Jahresumsatz von ca. € 7.000.000 etwa 80 Mitarbeiter.

Grundsätzlich sind mit CoDeSys all jene Automatisierungssysteme programmierbar, die nach dem internationalen Standard IEC 61131-3 programmiert werden können und mit CoDeSys kompatibel ausgeführt sind.

CoDeSys setzt sich aus 2 Programmen zusammen:

- CoDeSys Programmiersystem

Software zur Erstellung von Automatisierungsprojekten. (Sowohl Programmcode selbst, als auch zugehörige Visualisierungen)

- CoDeSys Laufzeitsystem „Control“

Mit CoDeSys Control kann auf einem Programmiergerät (PC) eine komplett lauffähige Steuerung simuliert werden, die über CoDeSys programmierbar ist. Dies bietet optimale Gegebenheiten für Tests des Programmes und der Visualisierung direkt am PC.

Mittlerweile arbeiten bereits mehr als 250 Gerätehersteller aus unterschiedlichen Branchen mit CoDeSys zusammen, bzw. setzen sie CoDeSys als Programmiertool für ihre Automatisierungsgeräte ein. Damit ist CoDeSys derzeit das meist genutzte hardwareunabhängige Entwicklungs- und Programmiersystem mit dem nach der IEC 61131-3 programmiert werden kann.

12 URL http://www.3s-software.com/index.shtml?de_homepage [15.12.10]

2.4.2. Prüfung auf Kompatibilität[13]

Im Zuge dieser Diplomarbeit soll überprüft werden, ob diverse Steuerungskomponenten der Firma B+R mit der Programmiersoftware CoDeSys kompatibel sind und damit durch diese konfigurierbar bzw. programmierbar sind.

Zunächst wurde im Rahmen dieser Arbeit die Kompatibilitätsprüfung anhand der Hersteller- bzw. Geräteliste der Firma 3S Smart Software, welche auf deren Internetpräsenz zur Verfügung gestellt wird, geprüft ob eine Kompatibilität mit Geräten der Firma Bernecker und Rainer vorliegt. Da in dieser Liste keine B+R-Geräte vorhanden sind, wurde die Firma 3S Smart Software noch zusätzlich per E-Mail kontaktiert, um eine etwas genauere Aussage in Bezug auf die Zusammenarbeit zwischen B+R und 3S Smart Software (CoDeSys) machen zu können.

Es folgt die Antwort per E-Mail der Firma 3S Smart Software[14]

„Sehr geehrter Herr Klotz,

vielen Dank für Ihre Anfrage.

Es gab zwar vor einigen Jahren eine Zusammenarbeit zwischen unserem Unternehmen und B&R. So wurde damals Teile von CoDeSys im Automation Studio eingesetzt. Diese Teile wurden aber von B&R komplett übernommen und wohl auch eigenständig weiterentwickelt.

Deswegen kann man heute die Steuerungen von B&R sicher nicht mit CoDeSys programmieren.

Allerdings stellt B&R auch Industrie-PCs her und bietet auf Kundenanfrage auch die CoDeSys SP RTE SoftSPS an, die dann wiederum mit CoDeSys V2.3 programmiert werden kann. Ich hoffe, diese Information hilft Ihnen weiter. Für tiefer gehende Fragen stehe ich Ihnen gern zur Verfügung.

Mit freundlichen Grüßen aus dem Allgäu,

Dipl. Ing. (FH) Roland Wagner I Marketing Manager"

Aus diesem E-Mail folgt also, dass grundsätzlich keine Kompatibilität der B+R Steuerungskomponenten mit der Programmier- und Entwicklungssoftware CoDeSys vorliegt. Ausgenommen davon ist die CoDeSys SP RTE SoftSPS, die von B+R auf Kundnwunsch auf deren Industrie-PC's installiert und ausgeliefert wird. Diese ist dann mittels CoDeSys V2.3 programmierbar.

13 URL http://www.3s-software.com/index.shtml?codesys_dev_dir [15.12.10]
14 E-Mail von DI FH Roland Wagner / Marketing Manager / Firma 3S Smart Software Solutions, r.wagner@3s-software.com, [17.12.10]

3. Programmiersoftware B+R

3.1. Allgemein[15]

Die integrierte Software-Entwicklungsumgebung von B+R nennt sich „Automation Studio“ und bietet Werkzeuge für alle Projektabschnitte eines Automatisierungsprojekts. In jeder Phase eines Projekts bietet Automation Studio also die optimale Schnittstelle zur Maschine bzw. zur Anlage.

Das bedeutet eine vollständige Integration von:

- Projektmanagement
- Programmierung
- Hardwaremanagement
- Integrierte Visualisierung
- Motion Control
- Diagnose
- Safety
- Simulation
- Hilfesystem
- Feldbusse
- Kommunikation
- Echtzeit Betriebssystem
- Fernwartung

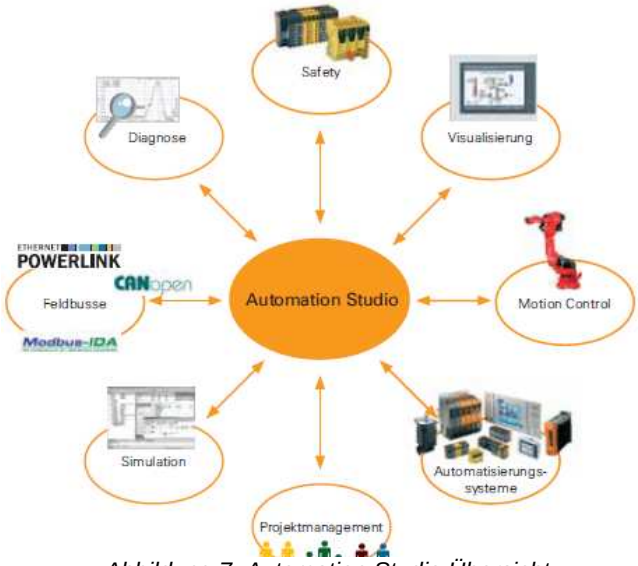

Abbildung 7: Automation Studio Übersicht

Die wichtigsten der in der Aufzählung enthaltenen Punkte werden im Folgenden näher erläutert.

3.2. Projektmanagement[16]

3.2.1. Projekt-Explorer

Die Kernkomponente des Projektmanagements stellen die 3 im Projektexplorer zur Verfügung gestellten Baumansichten dar.

3.2.2. Logische Ansicht (Logical View)

In der logischen Ansicht werden alle Softwareelemente in Form eines Baumes dargestellt. Jedes Objekt kann in einem eigenen Ordner als Paket verwaltet werden. Dabei kann jedes Paket noch in Softwarekomponenten und Dokumentationsdaten von Maschinenteilen oder Maschinenfunktionen gegliedert werden.

15 “Broschüre zu Automation Studio”, Bernecker und Rainer GesmbH, 2007;
16 “Broschüre zu Automation Studio”, Bernecker und Rainer GesmbH, 2007;

3.2.3. Konfigurationsansicht (Configuration View)

Die Konfigurationsansicht ermöglicht die Verwaltung von verschiedenen Hardware- und Softwarekonfigurationen. In jeder der Konfigurationen kann Hard- und Software vorkommen. Wird eine Configuration aktiviert, so wird diese auch in der physikalischen Ansicht angezeigt.

3.2.4. Physikalische Ansicht (Physical View)

In dieser Ansicht wird der aktive Hardwarebaum der ausgewählten Konfiguration angezeigt. Hier ist es möglich die Eigenschaften aller im Projekt enthaltenen Komponenten zu bearbeiten.

3.3. Versionsverwaltung[17]

Die Versionskontrolle ist ein weiterer wichtiger Bestandteil des Projektmanagements. Sie erlaubt das koordinierte, gleichzeitige Arbeiten mehrerer Personen am selben Projekt.

Das Ganze funktioniert über die Ablage des Projekts auf einem Server, was einen zentralen Zugriff mehrerer Personen auf dieselben Source-Dateien eines Projektes ermöglicht. Dabei werden Entwicklungsredundanzen durch gezieltes Reservieren bzw. Sperren von Dateien bewerkstelligt.

Die Änderungshistorie wird auf Dateiebene im Source-Code hinterlegt, wodurch die Möglichkeit besteht die Bearbeitung jeder Datei zu dokumentieren.

Um zu gewährleisten, dass die Projektstände reproduzierbar bleiben, wird bei jeder Änderung automatisch eine Erhöhung der Projektrevision durchgeführt.

Derzeit sind zur Revisionsverwaltung Schnittstellen zu den folgenden Systemen implementiert:

- Microsoft SourceSafe
- Subversion (SVN)

17 "Broschüre zu Automation Studio", Bernecker und Rainer GesmbH, 2007;

3.4. Programmierung[18]

Bei den Programmiersprachen unterscheidet man zwischen grafischen und textbasierten Sprachen. Automation-Studio bietet folgende Editoren zur Programmierung der Steuerungssysteme:

Grafische Editoren:	Kontaktplan, Ablaufsprache, Funktionsbausteinsprache
Textbasierte Editoren:	Anweisungsliste, Strukturierter Text, Ansi C,
	B+R Automation Basic

Die verfügbaren Programmiersprachen werden unter Punkt 4 im Detail behandelt.

3.5. Hardware-Management[19]

Alle I/O- oder Schnittstellenmodule können komfortabel an der jeweiligen Schnittstelle oder am lokalen I/O-Bus eingefügt und konfiguriert werden.

Weiters ist es möglich bei eingeschaltetem Monitor-Modus alle im Projekt konfigurierten I/O Module und deren einzelne Kanäle physikalisch zu testen und die Aktualwerte der jeweiligen Kanäle darzustellen. Ausgänge können hier, unabhängig zu ihren aktuellen Werten aus der Programmlogik, auf manuell eingegebene Werte gesetzt werden.

Jede Schnittstelle in der Hardwarekonfiguration kann über deren Eigenschaften eingestellt werden. Dies gewährleistet eine einheitliche und übersichtliche Art der Konfiguration.

3.6. Integrierte Visualisierung[20]

Mittels Automation Studio hat der User die Möglichkeit Visualisierungen zu den SPS-Programmen zu erstellen. Von einfachen Zeilendisplays bis hin zu integrierten oder abgesetzten XGA Display mit Tasten- oder Touchscreenbedienung können mit dem integrierten Editor alle Anforderungen an eine Visualisierung erfüllt werden.

Die wichtigsten Funktionen und Merkmale des Visualisierungstools werden in den folgenden Punkten erläutert und beschrieben.

3.6.1. Controls

Alle Steuerungselemente, die zur Erstellung von Prozessvisualisierungen notwendig sind, werden in der Visualisierungsumgebung von Automation Studio zur Verfügung gestellt. Sie können auf einfache Art und Weise mit Prozessvariablen verbunden werden, was eine zustandsgebundene Visualisierung von Prozessanlagen ermöglicht und somit die Arbeit mit den Steuerungssystemen für den User erheblich vereinfacht und den Bedienkomfort maßgeblich erhöht.

---

18 "Broschüre zu Automation Studio", Bernecker und Rainer GesmbH, 2007;
19 "Broschüre zu Automation Studio", Bernecker und Rainer GesmbH, 2007;
20 "Broschüre zu Automation Studio", Bernecker und Rainer GesmbH, 2007;

3.6.2. Property Sheets

Die diversen Komponenten eines Prozessbildes können mittels zugehörigem Property Sheet konfiguriert werden, was eine einfache und einheitliche Bedienung aller Visualisierungskomponenten ermöglicht.

3.6.3. Style Sheets

Mittels der Stylesheets kann das Erscheinungsbild jedes einzelnen Visualisierungsobjektes angepasst werden. Dadurch kann die Visualisierung optimal auf die Bedürfnisse des Users angepasst werden.

3.6.4. Bildebenen (Layer)

Mit Hilfe der Unterteilung eines Prozessbildes in mehrere Ebenen wird es möglich, diese beliebig miteinander zu kombinieren und passend zur Laufzeit zu animieren. Dadurch können wiederkehrende Bildinformationen zentral definiert werden und an neue Gegebenheiten angepasst werden.

3.6.5. Sprachen

Um dem späteren User ein Umschalten zwischen verschiedenen Sprachen der Visualisierung zu ermöglichen, ist ein Tool zum übersichtlichen Einsatz unterschiedlicher Sprachen im Visualisierungstool integriert. Damit ist es möglich Texte nach deren Sprache sortiert zu organisieren und während der Laufzeit der richtigen Sprache zuzuordnen.

3.6.6. Tasten

Durch das integrierte Tool zur Erstellung von Tasten-Mappings ist es zum Beispiel möglich eine Bildschirmtastatur zur Bedienung via Touch-Panel zu erstellen. Die logische Zuordnung der Tasten erfolgt im Tastenlayout, wobei es auch hier möglich ist die Tastenbelegung von der aktuell eingestellten Benutzersprache abhängig zu machen.

3.6.7. Farbpalette

Die Farbverwaltung ist für alle Prozessbilder gekoppelt und bewerkstelligt ein zentrales Verwalten des farblichen Erscheinungsbildes aller Elemente der Visualisierung. Dies bietet eine erhebliche Zeitersparnis, wenn der User ein eigenes Farbschema vorgibt und dieses einheitlich für die komplette Visualisierung umgesetzt werden soll.

3.6.8. Physikalische Einheiten

Alle Einheiten, die für die grafische Darstellung physikalischer Größen benötigt werden, sind in sinngemäßen Gruppen sortiert und vereinfachen somit den Umgang mit Werten aller Art.

3.6.9. Datenpunkte

Über die implementierte Datenpunktverwaltung kann eine Verbindung zwischen Prozessvariablen und Visualisierungsobjekten erstellt werden. Dies erlaubt die Erstellung von kompletten, laufzeitfähigen Prozessvisualisierungen ohne jeglichen Programmieraufwand zu betreiben.

3.6.10. Alarmsysteme

Über die Alarmsysteme des Visualisierungstools werden bestimmte Zustände des Systems aufgezeichnet und entsprechend auf diese Reaktionen im Programm ausgelöst. Alle Alarme können verschiedenen Alarmierungsgruppen zugeordnet werden, womit eine Unterteilung in Meldungen, Warnungen und Alarme möglich ist. Die unterschiedlichen Gruppen können innerhalb der Visualisierung auch in verschiedenen Farben präsentiert werden.

3.6.11. Trend System

Mit der Trend Control bietet Automation Studio ein multifunktionales Tool zur Erstellung von Trenddarstellungen in den Prozessvisualisierungen.

3.6.12. Terminal Mode

Für Bedienterminals können in Automation Studio eigene Visualisierungen erstellt werden, die unabhängig zur lokalen Visualisierung dargestellt werden. Es ist jedoch auch möglich, ein Abbild der auf dem lokalen Gerät laufenden Visualisierung auf dem Terminal darzustellen. Die Terminals können abgesetzt von Steuerungseinheiten und Visualisierungsrechnern betrieben werden.

3.6.13. VNC (Virtual Network Computing)

Über die Funktion des Virtual Network Computing ist es möglich eine Fernwartungsfunktion mit großem Funktionsumfang einzurichten. Dabei wird der Bildschirminhalt der Visualisierung auf einem lokalen Rechner angezeigt und die lokal durchgeführten Mausbewegungen bzw. Tastatureingaben auf den VNC-Server übertragen und somit für alle Teilnehmer sichtbar gemacht.

4. Programmiersprachen für B+R Steuerungssysteme

4.1. Die IEC 61131[21]

Die IEC 61131 befasst sich mit den Grundlagen zur Programmierung speicherprogrammierbarer Steuerungen. Der Inhalt des IEC Standards 61131 gliedert sich in folgende 5 Teile:

- Teil 1: Allgemeine Informationen
  Begriffsbestimmungen, Funktionsmerkmale einer speicherprogrammierbaren Steuerung

- Teil 2: Betriebsmittelanforderungen und Prüfungen
  Funktionelle, mechanische und elektrische Anforderungen, Umgebungebedingungen, Typenprüfungen, Beanspruchungsklassen

- **Teil 3: Programmiersprachen**
  Softwaremodell, Programmiersprachen

- Teil 4: Anwenderrichtlinien
  Systemanalyse, Wartung, Geräteauswahl

- Teil 5: Kommunikation
  Kommunikation und Vernetzung von Geräten diverser verschiedener Hersteller, Netzkommunikation, Verarbeitung von Alarmen, Netzwerkverwaltung und Kommunikation

In dieser Diplomarbeit ist ausschließlich Teil 3 der IEC 61131 relevant, da es gilt die Kompatibilität von Automation Studio mit den in der Norm enthaltenen Programmiersprachen zu bewerten.

21 URL http://control-net.fh-duesseldorf.de/IEC61131/iec_d/iec11.htm [05.12.10]

4.2. Programmiersprachen für B+R Steuerungssysteme nach IEC 61131

Um zu bewerten welche Programmiersprachen der IEC 61131 von den Systemen der Firma B+R unterstützt werden, ist also der Vergleich mit den in Teil 3 der IEC Norm 61131 enthaltenen Programmiersprachen notwendig.

Folgende 5 Programmiersprachen sind dort gelistet:

Englisch		Deutsch	
Abk.	**Bezeichnung**	**Abk.**	**Bezeichnung**
IL	Instruction List	AWL	Anweisungsliste
LD	Ladder Diagram	KOP	Kontaktplan
FBD	Function Block Diagram	FBS	Funktionsbausteinsprache
SFC	Sequential Function Chart	AS	Ablaufsprache
ST	Structured Text	ST	Strukturierter Text

Tabelle 3: Programmiersprachen der IEC 61131-3

Von diesen 5 Sprachen sind alle bis auf SFC, also die Ablaufsprache in Automation Studio integriert. Diese wurde in Automation Studio durch die ähnliche Sprache CFC ersetzt. Im Folgenden werden die in der IEC 61131 enthaltenen Programmiersprachen kurz erläutert und deren Unterschiede zueinander mittels eines einfachen Programmbeispiels aufgezeigt.

4.2.1. Die Anweisungsliste (Instruction List)[22]

Bei der AWL werden die Anweisungen wie in einer maschinenorientierten Programmiersprache (Assembler) untereinander geschrieben. Es handelt sich also um eine textuelle Programmiersprache. Jede der Anweisungen enthält einen Operator und einen Operand, im Idealfall werden diese noch durch einen sinngemäßen Kommentar ergänzt.
Da die AWL eine zeilenorientierte Programmiersprache ist, wird jede Ausführungsanweisung an die SPS in genau einer Zeile beschrieben.
D.h., eine Anweisung in der AWL setzt sich folgendermaßen zusammen:

Sprungmarke: **Operator** **Operand** *// Kommentar*

Bei der **Sprungmarke** handelt es sich um optional verwendbare Adressen, die verwendet werden können, um mittels eines Jump-Befehls von einer Stelle im Programm zu einer anderen zu springen. Dieser Teil der Anweisungszeile kann also entfallen und ist nicht notwendig.

22 "SPS-Programmierung mit IEC 61131-3", Karl Heinz John u. Michael Tiegelkamp, 2009

Der **Operator** ist jener Teil, der die gewünschte Operation enthält (z.B. AND, OR, GE, …). Beim **Operand** handelt es sich um jene Adresse in der Steuerung (z.B. Eingangsbit), die mittels des Operators in dieser Zeile behandelt wird. Ein **Kommentar** wird im Idealfall zu jeder Zeile hinzugefügt, da dies eine spätere Fehlersuche bzw. die allgemeine Orientierung innerhalb des SPS-Programmes erheblich erleichtern kann. Generell muss jedoch kein Kommentar vorhanden sein.

Ein Beispiel zur Programmierung mittels AWL befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

4.2.2. Der Kontaktplan (Ladder Diagram)[23]

Beim KOP handelt es sich um eine grafische Programmiersprache, die in etwa klassischen Stromlaufplänen nachempfunden ist. Links und rechts befindet sich in der LD-Programmieroberfläche jeweils ein Stromzweig. Dabei stellt der linke Stromzweig die Versorgungsspannung dar und der rechte Stromzweig einen Massepunkt. Zwischen diesen Stromzweigen werden die einzelnen Anweisungen in Form von Netzwerken eingefügt. Wie bei einem klassischen Schaltplan werden hier UND-Verknüpfungen in Serie dargestellt und ODER-Verknüpfungen als Parallelschaltung. Eine Anweisung innerhalb eines LD-Programmcodes kann also wie folgt aussehen:

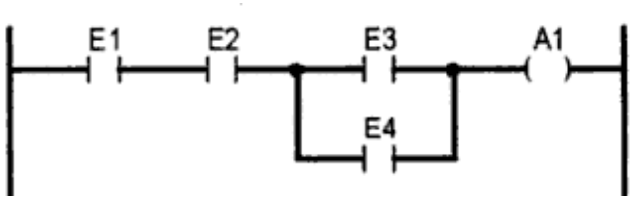

Abbildung 8: Beispiel Kontaktplan

Laut diesem Beispiel ist also A1 solange aktiv, solange E1 UND E2 UND (E3 ODER E4) aktiv sind.

Ein Beispiel zur Programmierung mittels LD befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

23 "SPS Programmierung nach IEC 61131-3", Heinrich Lepers, 2005

4.2.3. Die Funktionsbausteinsprache (Function Block Diagram)[24]

Die FBS ist eine grafik-basierte Sprache. Die Darstellung erfolgt mittels der Logiksymbole der booleschen Algebra. Dadurch ist diese Sprache auch für Laien sehr schnell und leicht zu erlernen und ist sehr gut zu verstehen. Besonders bei Verknüpfungssteuerungen bietet diese Programmiersprache eine gute Nachvollziehbarkeit.

Eine Anweisung innerhalb eines FBS-Programmes kann folgendermaßen aussehen:

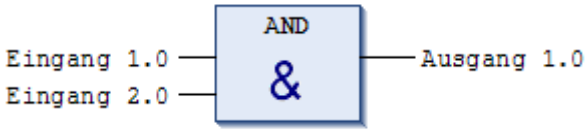

Abbildung 9: Beispiel Funktionsbausteinsprache

In oben abgebildetem Beispiel ist also der Ausgang 1.0 solange aktiv, solange Eingang 1.0 UND Eingang 2.0 aktiv sind. Ist einer der beiden Eingänge nicht mehr aktiv, so ist auch der Ausgang inaktiv.

Ein Beispiel zur Programmierung mittels FBS befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

4.2.4. Die Ablaufsprache (Sequential Function Chart)[25]

Auch hier handelt es sich um eine grafische Programmiersprache. Sie bietet große Vorteile in der anschaulichen Darstellung von Steuerungsabläufen und ist sowohl für zeit- wie auch ereignisorientierte Abläufe geeignet. Es wird eine Verkettung von Programmschritten dargestellt, die durch diverse Bedingungen zur Weiterschaltung verbunden sind.

Hier ein einfaches Beispiel zur Programmierung mittels SFC:

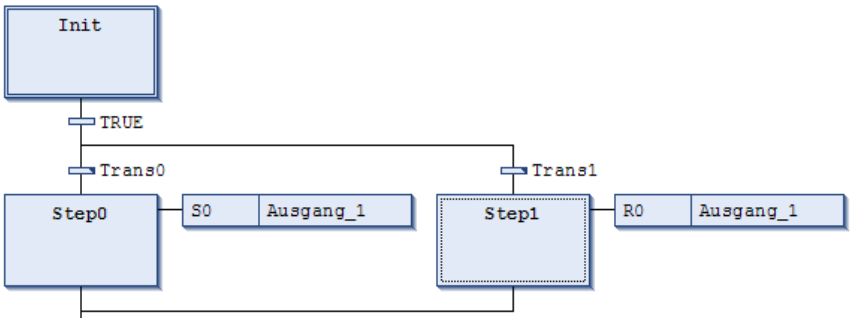

Abbildung 10: Beispiel Ablaufsprache

24 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm [05.12.10]
25 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm [05.12.10]

In diesem Beispiel wird Ausgang_1 gesetzt, wenn Trans0 wahr ist. Wird Trans1 wahr, so wird Ausgang_1 rückgesetzt.

Ein Beispiel zur Programmierung mittels AS befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

4.2.5. Der strukturierte Text (Structured Text)[26]

Der strukturierte Text ist wie die AWL eine textuelle Programmiersprache, jedoch handelt es sich hier um keine maschinenorientierte Sprache. Im Fall des ST spricht man eher von einer höheren Programmiersprache. Es können nämlich über abstraktere befehle als bei der AWL, wesentlich mächtigere Befehlskonstrukte erstellt werden. Der zu erstellende Programmablauf wird in einzelne Befehle zerlegt, wobei jede Anweisung aus einer Berechnung/Bestimmung des Wertes und dessen Zuweisung besteht. Die einzelnen Anweisungen werden im strukturierten Text durch Strichpunkte getrennt. Der Zeilenumbruch spielt im ST keine Rolle. Es ergibt sich also folgendes Beispiel für eine Anweisung innerhalb eines ST-Programmes:

A:= B (*Strecke_1*) + C (*Strecke_2*) + D (*Strecke_3*);

Bei A, B, C und D handelt es sich Variablen (Operanden), die im obigen Beispiel für eine Addition verwendet werden. Die Symbole (:=, +, ;) werden für die Operationen bzw. für den Abschluss einer Anweisung verwendet. Die Kommentare im ST werden einheitlich durch die Schreibweise (*Kommentar*) gekennzeichnet.

Ein Beispiel zur Programmierung mittels ST befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

4.3. Programmiersprachen für B+R Systeme (Nicht nach IEC 61131)[27]

Die Programmiersoftware unterstützt noch 3 weitere Programmiersprachen, die nicht in der IEC 61131 enthalten sind.

4.3.1. Automation Basic

Ähnlich dem Strukturierten Text, eignet sich diese Sprache vor allem für Anwender, die eine Sprache verwenden wollen, die einfach und schnell zu erlernen ist. Sie beinhaltet jedoch auch Funktionen und Befehle von Hochsprachen. Dadurch bietet Automation Basic eine gute Mischung aus höheren Programmiersprachen und der maschinenorientierten Programmierweise.

26 "SPS-Programmierung mit IEC 61131-3", Karl Heinz John u. Michael Tiegelkamp, 2009
27 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm [05.12.10]

Hier ein einfaches Beispiel zur Programmierung mittels Automation Basic:

```
PROGRAM _CYCLIC


if Eingang_1=true AND Eingang_2=true then
   Ausgang_1=true
   endif


END_PROGRAM
```

Abbildung 11: Beispiel Automation Basic

Laut dem Programmbeispiel ist also Ausgang_1 aktiv, solange Eingang_1 UND Eingang_2 aktiv sind.

Ein Beispiel zur Programmierung mittels Automation Basic befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

4.3.2. CFC (Continous Funktion Chart)[28]

Die Sprache CFC ähnelt sehr stark der Funktionsbausteinsprache. Jedoch wird in CFC nicht netzwerkorientiert (wie in FBS) gearbeitet, sondern der CFC ermöglicht es die Funktionsbausteine frei auf dem Bildschirm zu platzieren. Dadurch lassen sich zum Beispiel Rückkopplungen ohne Zwischenvariablen realisieren. Dies ermöglicht z.B. eine übersichtliche Darstellung von Applikationen. Ein einfaches Beispiel für die Programmierung mittels CFC ist unten folgend ersichtlich:

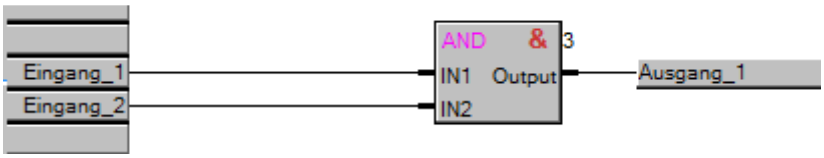

Abbildung 12: Beispiel Continous Function Chart

28 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm [05.12.10]

4.3.3. ANSI-C[29]

ANSI-C ist wohl die mächtigste Programmiersprache, die von B+R Systemen unterstützt wird. Es handelt sich um eine textbasiert Programmiervariante. Durch die Möglichkeit in ANSI-C Prozessvariablen und Funktionsblöcke der IEC Sprachen zu nutzen, wird dem Anwender ein sehr hochwertiges Werkzeug zur Programmierung geboten. Einzig die speziellen Zeichen der Logikverknüpfungen und Zuweisungen sind für SPS-Programmierer, die ansonsten in IEC konformen Sprachen programmieren, etwas gewöhnungsbedürftig.

Hier ein einfaches Beispiel zur Programmierung mittels ANSI-C:

Ausgang_1 = (Eingang_1 || Eingang_2) && Eingang_3

Im angegebenen Beispiel ist als Ausgang_1 aktiv, wenn Eingang_3 UND entweder Eingang_1 ODER Eingang_2 aktiv sind.

Ein Beispiel zur Programmierung mittels ANSI-C befindet sich in Punkt 6 / Programmierung und Simulation von Beispielen.

5. Kommunikationsverbindungen von B+R Steuerungssystemen

Um es zu ermöglichen, dass Steuerungssysteme verschiedener Hersteller miteinander kommunizieren können, ohne auf die herkömmliche verdrahtungstechnische Verbindung der Geräte zurückzugreifen, arbeitet man seit einiger Zeit an der Entwicklung von Feldbussen und zunehmend auch Industrial Ethernet. Es gibt heutzutage bereits ca. 50 verschiedene Feldbusse und mehr als 20 Industrial-Ethernet-Protokolle.

29 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm [05.12.10]

Folgende Kommunikationsstandards werden von Steuerungssystemen von B+R unterstützt und sind somit in „Automation Studio“ als Kommunikationsverbindungen parametrierbar:

- **POWERLINK – Ethernet TCP/IP**
- PROFINET
- Profibus DP
- DeviceNet
- CAN-Bus
- CANopen

Im Rahmen dieser Diplomarbeit wird Ethernet Powerlink am detailliertesten behandelt, da laut Vorgabe die Leistungsparameter und Besonderheiten dieses Netzwerkes aufzuzeigen sind. Bei den anderen Kommunikationsnetzen wurde lediglich eine Kurzbeschreibung erstellt, um sie grundlegend aufzuzeigen.

5.1. Powerlink – Ethernet TCP/IP

5.1.1. Entwicklung[30]

Ethernet Powerlink wurde ursprünglich von der Firma Bernecker und Rainer entwickelt und wird heute von der offenen Anwender- und Anbietergruppe EPSG (Ethernet Powerlink Standardization Group) als offener Standard weiterentwickelt und spezifiziert.

5.1.2. Allgemeines[31]

Powerlink ist ein streng deterministisches Echtzeitprotokoll auf der Basis von Fast-Ethernet (100 MBit).

Die Übertragung der zyklischen I/O und Motion Daten geschieht zeit-isochron, die Kommunikation zwischen den Netzwerkteilnehmern (Knoten) passiert jedoch asynchron, wofür ein Teil der Bandbreite reserviert ist.

30 “Automation Studio Content Help”, Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18
31 “Automation Studio Content Help”, Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

5.1.3. Technische Daten[32]

Übertragung:	Standard Fast Ethernet, IEEE 802.3u, 100Base-TX
Datenübertragungsrate:	100 Mbps
Kommunikationsarten:	zyklisch, azyklisch
Nettodaten:	min. 1 Byte, max. 1488 Byte (Powerlink V1) bzw. 1490 Byte (Powerlink V2)
Stationszahl:	Max. 253 (Powerlink V1), max. 239 (Powerlink V2) – ist jedoch abhängig vom Speicherausbau des Managing Node
Betriebsarten f. Stationen:	synchron oder asynchron zum Powerlink-Zyklus
Zykluszeiten:	parametrierbar in 1 µs-Schritten (min. 200 µs bis max 30-100ms)
Segmentlänge:	max. 100m bei Twisted Pair, entsprechend mehr bei Einsatz eines Lichtwellenleiters
Topologie:	Linie, Stern oder Baum
Kabel:	Twisted Pair, S/UTP, AWG26
	Kat. 5 Patch Kabel („Crossover cable“). Bei Typ-2 Modulen auch 1:1-Kabel verwendbar.
Watchdogfunktion:	Ja, Software und Hardware
Galvanische Trennung:	Bus und I/O

5.1.4. Arten von Nodes[33]

Geräte innerhalb eines Powerlink-Netzwerkes können entweder als Managing Node (MN) oder als Controlled Node (CN) konfiguriert sein. Es gibt jedoch pro Netzwerk genau einen MN, welcher die Abläufe im Netzwerk bestimmt.

Ob ein Powerlink-Gerät MN und CN sein kann, oder eine bestimmte Konfiguration besitzt, hängt vom Typ des Gerätes ab.

32 “Automation Studio Content Help”, Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18
33 “Automation Studio Content Help”, Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

5.1.5. Grundprinzip[34]

Das Grundprinzip von Ethernet Powerlink ist die zyklische Kommunikation, wobei dabei derselbe Kommunikationsablauf ständig wiederholt wird.

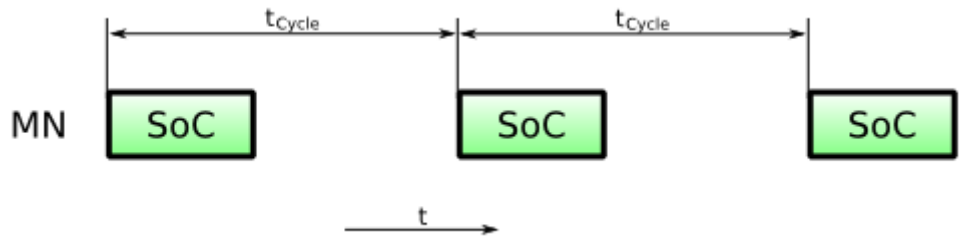

Abbildung 13: Powerlink, Start of Cycle

Wie in der obigen Abbildung ersichtlich wird zu Beginn jedes Zyklus eine „Start of cycle"-Nachricht (SoC) versandt. Der Versand erfolgt als Multicast und dient lediglich der Zeitsynchronisation.

5.1.6. Kompletter Zyklus[35]

Als Beispiel wird hier ein kompletter Zyklus eines Powerlink V1 Netzwerkes beschrieben, welcher folgendes Aussehen aufweist:

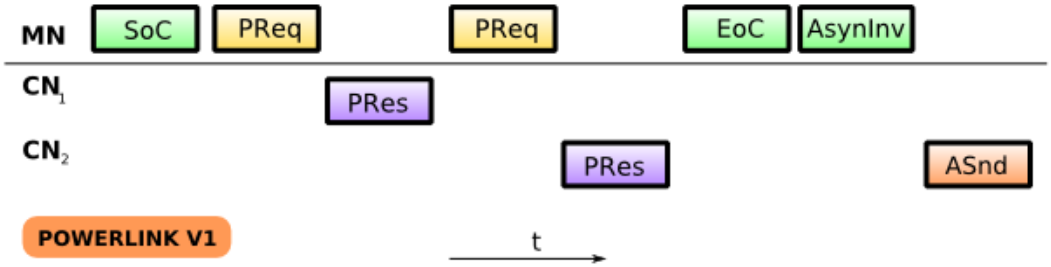

Abbildung 14: Powerlink, Zyklus V1

Beim **SoC** wird wie bereits beschrieben eine Multicast-Nachricht versandt, welche für alle Netzwerk-Teilnehmer die Zeit-Synchronisation darstellt.

Der **PReq** wird vom MN als Unicast direkt an einen CN adressiert versandt und enthält die Output-Daten, die nur für diesen CN bestimmt sind. Es werden also alle CNs hintereinander vom MN mit einem **PReq** angesprochen.

Die Antwort des CN erfolgt durch den **PRes** , welcher vom betroffenen CN als Multicast versandt wird. Dies ermöglicht, dass nicht nur die Input-Daten an den MN gesandt werden können, sondern auch Daten eines CN für alle anderen CNs zugänglich gemacht werden können. Dadurch können die Zeiten für den Austausch zwischen den Stationen erheblich reduziert werden, da sie nicht über den MN geschleust werden müssen.

---

34 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18
35 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

Durch das eben beschriebene System werden Kollisionen am Netzwerk vermieden und ein deterministisches Zeitverhalten erreicht.

Welche Daten in welcher Struktur in den **PReq**- und **PRes**- Frames enthalten sind, wird einmal im Hochlauf konfiguriert und ist im Anschluss fixiert. Die Werte innerhalb eines Frames werden durch einen Offset gekennzeichnet, anhand dessen sie von den anderen CN's ausgelesen werden können. In jedem Zyklus steht ein vollständig aktualisiertes Prozessabbild zur Verfügung, da es dauernd aktualisiert wird. Dadurch ergit sich ein sehr robustes Systemverhalten, da bei einer Übertragungsstörung im darauffolgenden Zyklus wieder alle Daten vollständig und aktuell übertragen werden.

Die beiden Nachrichten **EoC** und **AsynInv** dienen dem MN dazu den CNs mitzuteilen, dass die Übertragung der zyklischen Daten beendet ist und welche Station asynchrone Daten senden darf. Pro Zyklus kann nur ein asynchrones Frame übertragen werden.

5.1.7. Powerlink V1 / Powerlink V2[36]

Für die Übertragung innerhalb eines Powerlink V2 Netzwerkes ergeben sich folgende zusätzliche Features im Vergleich zu einem Powerlink V1 Netzwerkes:

- Unterstützung von Safety-Produkten
- Einbingung von Fremdgeräten mithilfe von XML Beschreibungsdateien (.XDD)
- Konfigurierbare Größe des asynchronen Kanals – unterstützung von Streaming
- PollResponse MN
- PollResponse Chaining Mode
- Laufende Performance-Optimierung

Powerlink V2 ist zu 100 % abwärtskompatibel zu Powerlink V1. Es können aber auch alle neuen B+R Module in einem Powerlink V1 Netzwerk betrieben werden.

36 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

5.1.7.1. Poll Response Chaining Mode[37]

Mit Hilfe von Poll Response Chaining ist es möglich, die Anzahl der in einem POWERLINK-Zyklus möglichen Stationen zu erhöhen.

Anhand der folgenden Bilder kann der Unterschied zwischen Übertragung ohne Chaining-Funktion und Übertragung mit Chaining-Funktion anschaulich dargestellt werden:

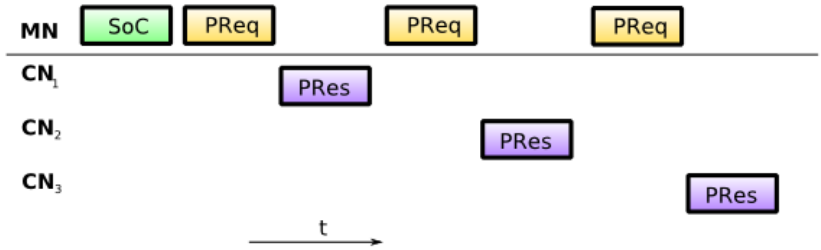

Abbildung 15: Powerlink, ohne Chaining Mode

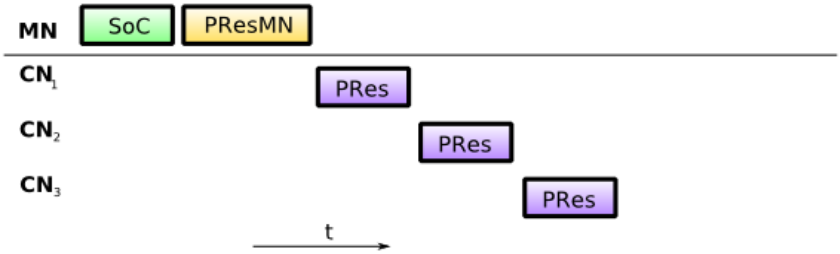

Abbildung 16: Powerlink, mit Chaining Mode

Poll Response Chaining bietet die Möglichkeit, ein POWERLINK-Netzwerk in mehrfacher Hinsicht zu optimieren:

- Durch den Wegfall der Poll Request-Frames ist es möglich, eine größere Anzahl von Stationen in der gleichen Zykluszeit unterzubringen.
- Durch Poll Response Chaining werden mehr Stationen in der ersten Hälfte des POWERLINK-Zyklus angesprochen, was die Verwendung der Reaktionszeitoptimierung Minimal input latency an der Taskklasse erleichtert.

37 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

5.1.7.2. Poll Response MN[38]

Hier besteht für den MN optional die Möglichkeit, einen Poll Response (PResMN) zu versenden. Damit kann auch der MN, gleich wie die CNs, Daten an die anderen Teilnehmer übertragen

5.1.8. Adressierung[39]

Jeder Teilnehmer hat eine Knotennummer, welche meist über einen Nummernschalter eingestellt wird.

Der folgenden Tabelle sind die zulässigen Wertebereiche zu entnehmen:

	POWERLINK V1	POWERLINK V2
MN	0	240
CN	1 .. 253	1 .. 239
Reserviert	254, 255	0, 241 .. 255

Tabelle 4: Powerlink, Wertebereiche Knotennummern

5.1.9. Diagnose[40]

Da Ethernet Powerlink auf standardkonformem Ethernet aufsetzt, können alle dafür verfügbaren Diagnosewerkzeuge verwendet werden. Dies stellt einen großen Vorteil in Bezug auf die Wartung eines Powerlink-Netzwerkes dar.

38 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18
39 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18
40 "Automation Studio Content Help", Bernecker und Rainer GesmbH, Automation Studio V3.0.81.18

5.1.10. Marktstudie zu Ethernet Powerlink

Folgende Marktstudie zu Ethernet Powerlink soll einen Überblick über die aktuellen Marktanteile der verschiedenen Industrial-Ethernet-Systeme bieten und den Marktanteil des Ethernet Powerlink Systems am Markt verdeutlichen.

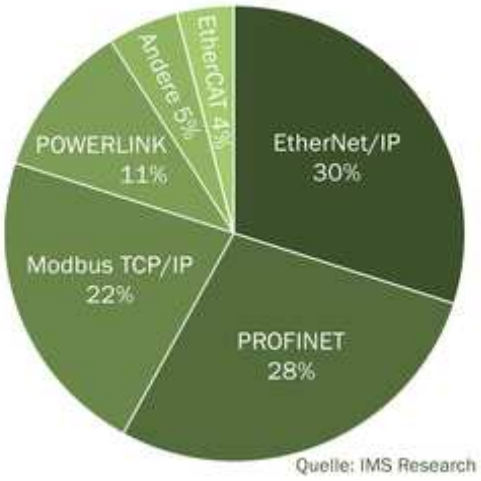

Abbildung 17: Powerlink, Marktanteil

"Wo Maschinen- und Sensordaten in harter Echtzeit übertragen werden, zählt POWERLINK neben PROFINET zu den meistgenutzten Industrial-Ethernet-Protokollen. Laut einer aktuellen Studie des unabhängigen Marktforschungsinstituts IMS Research zur internationalen Marktverbreitung industrieller Ethernet-Systeme erreicht POWERLINK einen Marktanteil von elf Prozent. Damit belegt POWERLINK in der Gesamtbetrachtung nach Ethernet/IP, PROFINET und Modbus TCP/IP den vierten Platz; allerdings eignen sich EtherNet/IP und Modbus TCP/IP nicht zur schnellen Antriebssteuerung."

"Zudem subsumiert die Studie im Profinet-Marktanteil von 28 Prozent sämtliche Profinet-Varianten, von denen ein großer Teil zu den nicht echtzeitfähigen Varianten des Siemens-Protokolls zählt. Den Erfolg von POWERLINK sieht Stefan Schönegger, Business Manager Open Automation Technologies bei B&R, in verschiedenen Faktoren begründet: „Als B&R die POWERLINK-Spezifikation veröffentlichte, war es das erste Echtzeitprotokoll, mit dem sich Kommunikationszyklen im Mikrosekundenbereich erreichen ließen. Damit wurde eine Nachfrage erfüllt, die in der Industrie gerade erst entstand und in der Folge zu einem rasanten Wachstum führte. Weitere Pluspunkte liegen in der Herstellerunabhängigkeit, der Nähe zum Ethernet-Standard und der Leistungsbandbreite: So lassen sich mit POWERLINK vom Antrieb bis zur Leitebene durchgängige Anwendungsstrukturen schaffen, die alle denkbaren Netzwerkkomponenten integrieren. Darüber hinaus können heterogene Netzwerkarchitekturen realisiert werden, denen andere Kommunikationssysteme unter- oder übergeordnet sind." Wettbewerbsvorteile von POWERLINK ergeben sich heute zudem durch die Open-Source-Lizenz und die Unterstützung von Entwicklern und Anwendern durch die EPSG."[41]

41 URL http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/cookies_allowed.htm?caller=news_14230_DEU_HTML.htm [29.11.10]

5.2. Profinet[42]

Abbildung 18: Logo Profinet

Das Profinet wurde von Siemens in Zusammenarbeit mit der Profibus-Nutzerorganisation entwickelt. Die Basis des Profinet ist das bewährte Funktionsmodell des Profibus DP, jedoch wird als physikalisches Übertragungsmedium die Fast-Ethernet-Technologie genutzt. Profinet ist auf die schnelle Übertragung von I/O-Daten zugeschnitten und bietet trotzdem Möglichkeiten zum zeitgleichen Transport von Bedarfsdaten, Parametern sowie allgemeine IT-Funktionen. Wie beim Profibus DP werden die im Netzwerk enthaltenen Feldgeräte mittels einer Gerätebeschreibung (GSD-Datei) in das Projektierungstool eingebunden. In dieser Datei sind sämtliche Eigenschaften des Feldgerätes enthalten. Ein weiterer Unterschied zum Profibus DP ist, dass im Gegensatz zum Master-Slave-Verfahren ein Producer-Consumer-Modell verwendet wird, welches die Kommunikationsbeziehungen zwischen den gleichberechtigten Teilnehmern am Ethernet unterstützt.

5.3. Profibus DP[43]

Abbildung 19: Logo Profibus

Profibus steht für „Process Field Bus“ und ist ein universeller Feldbus, der sehr breite Anwendung in Fertigungs-, Prozess-, und Gebäudeautomatisierung findet. Entwickelt wurde der Profibus in einer Kooperation der Firma Siemens und der Profibus-Nutzerorganisation und ist in der internationalen Normenreihe IEC 61158 standardisiert. Er basiert auf international definierten Standards, wobei sich die Architektur des Protokolls am OSI Referenzmodell (Open System Interconnection) orientiert. In diesem sind für jeden Übertragungslayer genaue Aufgaben definiert. Layer 1 (Physical Layer) beschreibt die komplette Übertragungsphysik, in Layer 2 (Data Link Layer) wird das Buszugriffsprotokoll definiert und in Layer 7 (Application Layer) werden die Anwendungsfunktionen festgelegt. Durch die Festlegung der technischen Eigenschaften eines seriellen Bussystem, erlaubt es der Profibus, Automatisierungsgeräte von von der Feld- bis hin zur Zellenebene miteinander zu vernetzen. Da es sich beim Profibus um ein Multi-Master-System handelt, ist es möglich mehrere Automatisierungs-, Engineering- oder Visualisierungssysteme mit dezentralen Peripheriegeräten an einem Bussystem zu betreiben. Dabei stellt der Profibus einen Kommunikationsstandard dar, der eine

42 URL http://www.anybus.de/technologie/ethernet.shtml [03.12.10]
43 URL http://feldbusse.de/Profibus/profibus.shtml [03.12.10]

Kommunikation von Geräten verschiedener Hersteller ohne besondere Schnittstellen- und Konfigurationsanpassungen gewährleistet. Der Profibus ist sowohl für schnelle und zeitkritische Anwendungen, wie auch für komplexe Kommunikationsaufgaben in den verschiedensten Industrieanwendungen geeignet.

5.4. DeviceNet[44]

Abbildung 20: Logo DeviceNet

DeviceNet entstand aus einem Projekt von Rockwell in Unterstützung durch die ODVA (Open DeviceNet Vendor Association) und basiert als offener Feldbus-Standard auf dem CAN-Protokoll. Standardisiert ist das DeviceNet in der EN 50325 und die Pflege bzw. Spezifikation obliegt der ODVA. Das DeviceNet gehört zur Gruppe der CIP-basierten (CIP steht für Common Industrial Protocol) Netzwerke, welche auch das ControlNet und Ethernet/IP beinhaltet. Diese 3 Netzwerke sind folglich sehr gut aufeinander abgestimmt und bergen den großen Vorteil, dass für jede Ebene eines Kommunikationssystems ein entsprechendes Netzwerk zur Verfügung steht. Dadurch ergibt sich ein abgestuftes Netzwerk, das folgendermaßen aussieht:

- Ethernet/IP auf Leitebene
- ControlNet auf Zellenebene
- DeviceNet auf Feldebene

Bei DeviceNet handelt es sich um ein objektorientiertes Bussystem, das nach dem Producer/Consumer Verfahren arbeitet. DeviceNet-Geräte können also Client (Master) oder Server (Slave) sein, wobei jeder Client und jeder Server sowohl Producer als auch Consumer oder beides sein kann.

44 URL http://feldbusse.de/DeviceNet/devicenet.shtml [03.12.10]

5.5. CAN-Bus[45]

Abbildung 21: Logo CAN

Die CAN-Technologie ist seit 1994 standardisiert und wird hauptsächlich in der ISO 11898-1 beschrieben. Er stellt eine serielle Kommunikationstechnologie dar, welche hauptsächlich für einen sehr zuverlässigen Datenverkehr zwischen elektronischen Steuergeräten in Automobilen eingesetzt wird. In Bezug auf das OSI-Referenzmodell verwendet der CAN-Bus lediglich die Layer 1 und 2 (Physical Layer und Datalink Layer). Da der CAN-Bus eine hohe Verfügbarkeit sicherstellen muss, greift man auf eine Multi-Master-Architektur zurück. Das heißt, dass jeder Knoten jederzeit das Recht hat auf einen ohne Abstimmung mit den anderen Teilnehmern auf den Bus zuzugreifen. Dadurch ergibt sich die Gefahr, dass mehrere Knoten gleichzeitig auf den Bus zugreifen und dadurch Kollisionen entstehen. Um diese Kollisionen zu vermeiden ist im CAN-Bus CSMA/CR integriert (Carrier Sense Multiple Access / Collision Resolution). Über eine Bit-Arbitrierung wird dabei jeder Teilnehmer mit einer Priorität versehen. Versuchen nun mehrere Teilnehmer zum gleichen Zeitpunkt auf den Bus zuzugreifen, so darf lediglich der Teilnehmer mit der höchsten Priorität das Medium belegen. Dadurch ergibt sich für die anderen Teilnehmer nahezu keine Verzögerung der Übertragung und die Nachricht mit der höchsten Priorität kann fast verzögerungsfrei gesendet werden.

Üblicherweise liegt einem CAN-Bus eine Linientopologie zu Grunde. Das heißt, die Teilnehmer werden mittels Stichleitungen an den Bus angekoppelt, was ein Einbinden eines neuen Busteilnehmers sehr einfach macht. Die Übertragung basiert auf einer Spannungsdifferenz, wodurch beispielsweise in der Automobilindustrie Störspannungen, welche durch Motoren, Zündanlagen oder andere Bauteile induziert werden, unschädlich gehalten werden. Folglich gibt es beim CAN-Bus eine CAN-High und eine CAN-Low Leitung.

45 URL http://www.vector-elearning.com/vl_einfuehrungcan_de.html [01.12.10]

5.6. CANopen[46]

Abbildung 22: Logo CANopen

CANopen ist eine mittlerweile über weite Strecken verbreitete Anwendungsschicht des CAN. CANopen wurde vom CiA (CAN in Automation) Verband entwickelt und ist inzwischen in der internationalen Normierung angenommen worden. CANopen setzt sich aus den Kommunikations- und den Geräteprofilen zusammen. Die schnelle Verarbeitung von Ein- und Ausgangsdaten wird durch PDO's (Prozessdatenobjekte) gewährleistet. Sämtliche Prozessdaten und Geräteparameter sind in einem Objektverzeichnis strukturiert. Um auf die Daten im Objektverzeichnis zugreifen zu können, gibt es sogenannte SDO's (Servicedatenobjekte). Außerhalb dieser Standardobjekte gibt es zusätzlich noch ein paar spezielle Objekte innerhalb von CANopen, welche zum Beispiel für Netzwerkmanagement, Synchronisation oder Fehlermeldungen verwendet werden.

Für die oben angesprochenen Prozessdatenobjekte, also die Verarbeitung von Ein/Ausgangsdaten sind in CANopen mehrerere Kommunikationsarten möglich:

- Ereignisgesteuert (change of state): Eine Kommunikation findet statt, sobald eine Änderung der Daten stattgefunden hat. Es wird nicht das gesamte Prozessabbild kommuniziert, sondern nur jene Daten, die sich geändert haben.
- Zyklisch synchron (cyclic): Hier werden die Netzwerkteilnehmer über ein SYNC Telegramm dazu veranlasst, die neuen Ausgangsdaten zu übernehmen und mit ihren aktuellen Eingangsdaten zu antworten.
- Angeforderte Kommunikation (gepollt): Die Baugruppen werden über ein Datenanforderungstelegramm aufgefordert ihre Eingangsdaten zu senden.

Wie auch der CAN-Bus, basiert auch CANopen meist auf einer Linientopologie. Auch das CSMA (Carrier Sense Multiple Access) ist ein Merkmal, das CAN-Bus und CANopen verbindet. Da alle Teilnehmer am Bus jederzeit auf den Bus zugreifen können, besitzen alle Teilnehmer eine Priorität, die in Form eines Identifiers mit den Daten mitgesandt wird. Um also Kollisionen am Bus zu vermeiden, findet eine Busarbitrierung statt. Im Zuge der Arbitrierung wird die Bandbreite der Übertragung in Reihenfolge der Prioritäten vergeben, wodurch am Ende der Arbitrierungsphase immer nur ein Teilnehmer den Bus belegt. Somit werden Kollisionen vermieden und die zur Verfügung stehende Bandbreite wird für eine schnelle Datenübertragung optimal genutzt.

---

46 URL http://infosys.beckhoff.com/index.php?content=../content/1031/fc510x/HTML/CO_comFieldbus.htm&id= [01.12.10]

6. Programmierung und Simulation von Beispielen

6.1. Das Testsystem

In den folgenden Punkten werden die von B+R für diese Diplomarbeit zur Verfügung gestellten Steuerungskomponenten beschrieben.

6.1.1. Zentraleinheit

6.1.1.1. Allgemeines[47]

Bei der Zentraleinheit handelt es sich um einen Controller des Typs CP 3485-1. Dieser weist trotz seiner sehr kompakten Bauweise eine sehr hohe Leistungsfähigkeit auf. Dadurch bietet er sich zum Einsatz im Bereich von Standardanwendungen bis hin zu Applikationen mit hohen Performance-Ansprüchen an.

6.1.1.2. Technische Daten[48]

Prozessor:	Celeron 400
Schnittstellen:	1 x Ethernet, 1 x Powerlink V1/V2, 2 x USB, 1 x X2X Link, 1 x RS232 serielle Schnittstelle
Standardspeicher:	SDRAM 64 MB, SRAM 1 MB
Schnellste Taskklassen- Zykluszeit:	400 µs
Typische Befehlszykluszeit:	0,015 µs
Datenpufferung Lithium Batterie:	mind. 3 Jahre
Batterie Überwachung:	ja
Modulare Schnittstellen-Steckplätze:	3
Statusanzeigen:	CPU Funktion, Übertemperatur, Ethernet, Powerlink, CompactFlash, Batterie
Betriebstemperatur:	waagrechte Einbaulage 0 bis +55℃, senkrechte Einbaulage 0 bis +50℃
Kühlfunktion:	kein Lüfter, ausschließlich über Kühlkörper
Luftfeuchtigkeit:	5-95%, nicht kondensierend
Schutzart:	IP20

47 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;
48 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

6.1.1.3. Bedien- und Anschlusselemente[49]

Die folgende Grafik zeigt die im Zuge dieser Diplomarbeit eingesetzte Zentraleinheit inklusive einer Übersicht über die diversen Bedien- und Anschlusselemente der CPU 3485-1.

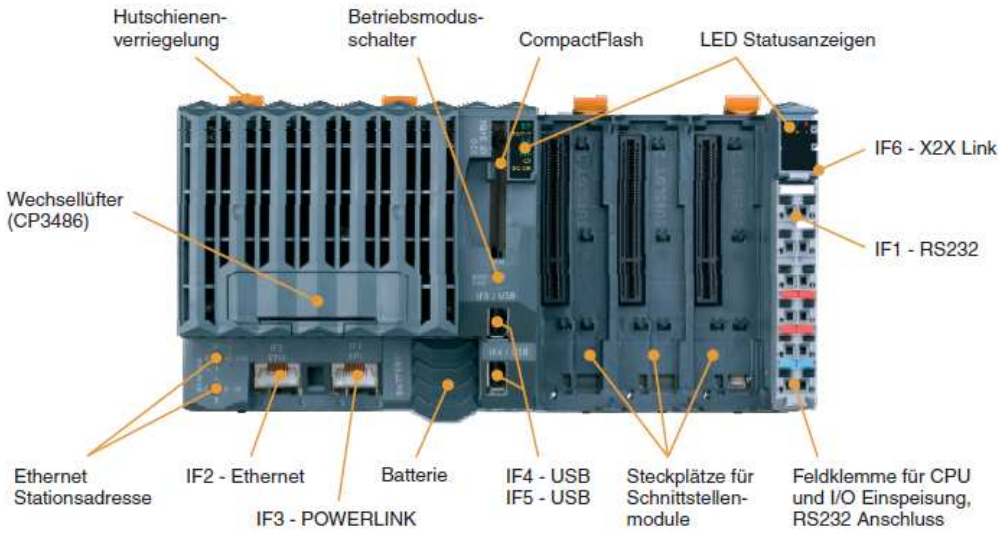

Abbildung 23: Bedien- / Anschlusselemente X20

6.1.2. Kommunikationsmodule[50]

6.1.2.1. X20IF1063-1 (Profibus Schnittstellenmodul)

X20 Schnittstellenmodul, 1 Profibus DP Slave Schnittstelle, max. 12 MBit/s, Potenzialtrennung integriert

6.1.2.2. X20BC0083 (Powerlink Schnittstellenmodul)

X20 Bus Controller, Powerlink V1/V2 Schnittstelle, integrierter 2-fach Hub, 2 x RJ45 Anschluss

6.1.2.3. X20IF1053-1 (DeviceNet Schnittstellenmodul)

X20 Schnittstellenmodul für DTM Konfiguration, 1 x DeviceNet Slave (Adapter) Schnittstelle, Potenzialtrennung integriert

49 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;
50 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

6.1.3. Ein- und Ausgangsbaugruppen[51]

6.1.3.1. X20DI4371 (digitales Eingangsmodul)

X20 digitales Eingangsmodul, 4 Eingänge, 24 VDC, Sink, Eingangsfilter parametrierbar, 3-Leitertechnik

6.1.3.2. X20DO4322 (digitales Ausgangsmodul)

X20 digitales Ausgangsmodul, 4 Ausgänge, 24 VDC, 0,5 A, Source, 3-Leitertechnik

6.1.3.3. X20AI2622 (analoges Eingangsmodul)

X20 analoges Eingangsmodul, 2 Eingänge, +/-10V oder 0…20mA oder 4…20mA, 12 Bit Auflösung, Eingangsfilter parametrierbar

6.1.3.4. X20AO2622 (analoges Ausgangsmodul)

X20 analoges Ausgangsmodul, 2 Ausgänge, +/-10V oder 0…20mA oder 4…20mA, 12 Bit Auflösung

6.1.4. Feldklemmen

6.1.4.1. X20TB06

X20 Feldklemme, 6-polig, 24V kodiert

6.1.4.2. X20TB12

X20 Feldklemme, 12-polig, 24V kodiert

6.1.5. Busmodule

X20BM11, X20 Busmodul, 24V kodiert, interne I/O-Versorgung durchverbunden

51 "Anwenderhandbuch System X20", Bernecker und Rainer GesmbH, 2009;

6.2. Programmbeispiele

In den nun folgenden Programmbeispielen wird versucht, die Programmierung von speicherprogrammierbaren Steuerungen mittels Automation Studio von B+R anhand von Beispielen in allen verfügbaren Programmiersprachen aufzuzeigen.

Beim Großteil der Beispiele handelt es sich um teilweise abgeänderte Übungen aus den Vorlesungen des Faches „Industrielle Steuerungen“ aus dem 3. Semester, welches durch Prof. Dr.-Ing. Römer unterrichtet wurde.

Die Sensorik bzw. Aktorik der jeweiligen Automatisierungsprojekte sind in beinahe allen Beispielen ausschließlich mit der Visualisierung gekoppelt, da dadurch eine Simulation und das Testen der Programmbeispiele am PC möglich war. Lediglich im Beispiel „Beispiel Feldgeräteanbindung“ kam eine X20-Steuerung von B+R in Kombination mit diversen Feldgeräten zum Einsatz, um das Anbinden von Sensoren und Aktoren an das Programm, bzw. die Koppelung von Ein- und Ausgängen mit Variablen im Programm zu beschreiben.

Um eine Test- bzw. Simulationsumgebung für die angeführten Beispiele zu schaffen wurden folgende Features von B+R genutzt:

- AR000

  Eine virtuelle Steuerung, auf die die erstellten Programme übertragen werden können und direkt am PC in den Betriebsmodus RUN versetzt werden können, um das gesamte Automatisierungsprogramm zu testen.

- Virtuelle Panels

  Ein Feature von Automation Studio, um virtuelle Bedienpanels zu erstellen, die auf einem PC oder Notebook ein Touch-Pad zur Bedienung eines Automatisierungsprojektes zur Verfügung stellen.

Folgendes Feature ist als Freeware verfügbar und wurde im Zuge dieser Diplomarbeit auch verwendet:

- Real VNC[52]

VNC steht für Virtual Network Computing. Diese Freeware erlaubt es dem User von einem beliebigen PC aus andere PC’s zu überwachen, bzw. Zu bedienen. Während dieser Arbeit wurde das Programm genutzt, um damit auf die mittels Automation Studio erstellten virtuellen Bedien-Panels zuzugreifen, die über eine in Automation Studio vorkonfigurierte IP-Adresse erreicht werden können.

52 URL http://www.chip.de/downloads/RealVNC_12997724.html [18.12.2010]

6.2.1. Einfache Transportbandsteuerung

6.2.1.1. Programmiersprache

Dieses Beispiel wurde mittels CFC (Continous Function Chart) erstellt.

6.2.1.2. Funktionsbeschreibung

Das Programm wurde erstellt um einen Pakettransport auf einem Förderband zwischen 2 Stationen zu ermöglichen. Station Links und Station Rechts sind mit einer Lichtschranke und einem Taster ausgestattet. Wenn die Lichtschranke der Station Links unterbrochen ist (Paket vorhanden) und die Lichtschranke der Station Rechts nicht unterbrochen ist, soll bei Betätigen des Tasters Links mit einer Anlaufverzögerung von 10 Sekunden der Rechtslauf des Transportbandes gestartet werden. Kommt das Paket bei Station Rechts an (Lichtschranke Rechts unterbrochen) soll das Transportband wieder zum Stillstand gebracht werden. Selbige Funktion wurde auch für die andere Richtung realisiert.

6.2.1.3. Allgemeines

Die beiden Taster und die beiden Lichtschranken wurden im erstellten Automatisierungsprojekt durch Buttons auf einem virtuellen Panel ersetzt. Die Ausgänge für Rechts- und Linkslauf des Förderbandes werden ebenfalls auf dem virtuellen Panel visualisiert und zu Simulationszwecken animiert.

6.2.1.4. TAG-Liste

Project: Transportband_NEU Transportband.var

Variable Declaration : C:\Projekte\Transportband_NEU\Logical\Transportband\Transportband.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Transportband * File: Transportband.var * Author: KLOSP * Created: January 08, 2011 ********** * Local variables of program Transportband **********		
Taster_Rechts	BOOL	Taster der rechten Bedienstation
Lichtschranke_Rechts	BOOL	Lichtschranke zur Paketerkennung der rechten Bedienstation
Lichtschranke_Links	BOOL	Lichtschranke zur Paketerkennung der linken Bedienstation
Taster_Links	BOOL	Taster der linken Bedienstation
Rechtslauf_EIN	BOOL	Rechtslauf des Transportbandes
Merker_Linkslauf_EIN	BOOL	Merker zur Aktivierung des Rechtslaufes
Linkslauf_EIN	BOOL	Linkslauf des Transportbandes
Merker_Rechtslauf_EIN	BOOL	Merker zur Aktivierung des Linkslaufes

Tabelle 5: TAG-Liste Transportbandsteuerung

6.2.1.5. Programmcode

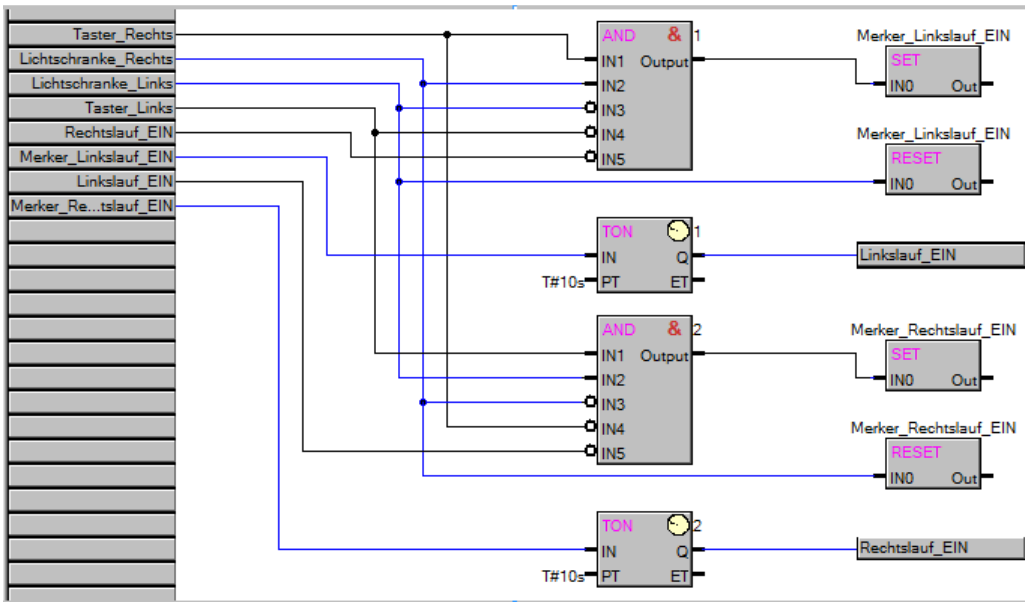

Abbildung 24: Programmcode Transportbandsteuerung

6.2.1.6. Visualisierung

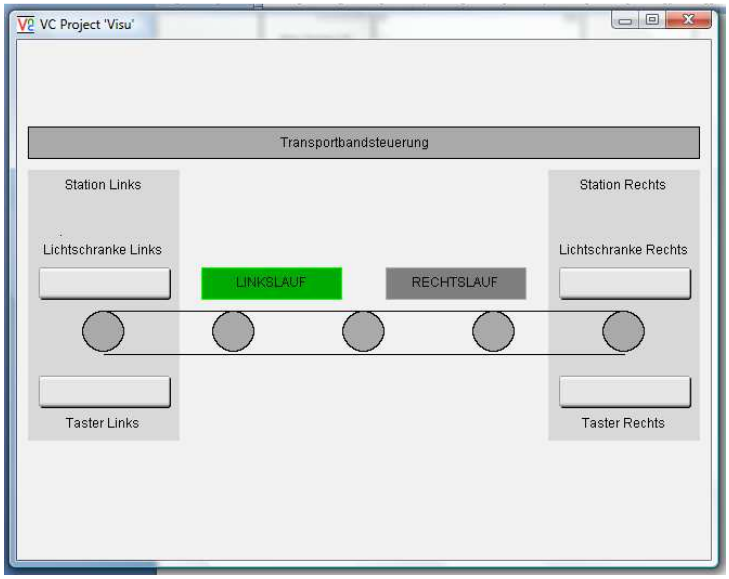

Abbildung 25: Visualisierung Transportbandsteuerung

6.2.2. Reaktorsteuerung

6.2.2.1. Programmiersprache

Die Reaktorsteuerung wurde in der Programmiersprache Function Block Diagram (Funktionsbausteinsprache) umgesetzt.

6.2.2.2. Funktionsbeschreibung

Der Reaktorablauf soll nach Betätigen des Tasters Start beginnen. Der erste Arbeitsschritt ist das Einfüllen von Flüssigkeit über das Zulaufventil. Zeitgleich zum Öffnen des Ventils soll das Rührwerk eingeschaltet werden. Der Stoff, der während der Reaktion in der Flüssigkeit gelöst werden soll, wird auf einem Förderband in der richtigen Dosierung bereit gestellt. Nach dem Befüllen des Reaktors mit der Flüssigkeit (also bei Erreichen des Endschalters Niveau MAX), soll der Förderbandmotor für 1 Minute eingeschaltet werden, um den Feststoff zuzuführen. Während das Rührwerk weiterhin läuft, soll nach dem Zuführen des zu lösenden Feststoffes die Heizung aktiviert werden, bis eine Innentemperatur von 80℃ erreicht wird.

Ist die Temperatur von 80℃ erreicht, so wird das Abflussventil geöffnet bis der Endschalter Niveau MIN anspricht. Daraufhin wird das Abflussventil geschlossen und die Reaktion ist beendet. Der nächste Reaktionsvorgang kann nach der Vorbereiten des Feststoffes wieder durch den Taster Start gestartet werden.

6.2.2.3. Allgemeines

Der Taster Start, die beiden Endschalter (Niveau MAX und Niveau MIN) wurden im Programm durch Buttons auf dem virtuellen Panel ersetzt. Die Innentemperaturmessung des Reaktors wurde als analoges Eingabefeld realisiert und die benötigten digitalen Ausgänge (Rührwerk, Förderband, Zulaufventil, Abflussventil und Heizung) wurden am virtuellen Panel visualisiert.

6.2.2.4. TAG-Liste

Project: Reaktor Reaktor.var

Variable Declaration : C:\Projekte\Reaktor\Logical\Reaktor\Reaktor.var

Name	Type	Description [1]
(* COPYRIGHT -- * Program: Reaktor * File: Reaktor.var * Author: KLOSP * Created: December 15, 2010 * Local variables of program Reaktor		
Temperatur	INT	Innentemperatur des Reaktors
Merker_Temperatur	BOOL	Merker für das Erreichen der vorgegebenen Innentemperatur
Taster_Start	BOOL	Taster zum Starten des Reaktionsvorganges
Zulaufventil_AUF	BOOL	Zulaufventil des Reaktors
Niveau_max	BOOL	Endschalter Niveau MAX
Abflussventil_AUF	BOOL	Abflussventil des Reaktors
Ruehrwerk_EIN	BOOL	Rührwerk des Reaktors
Niveau_min	BOOL	Endschalter Niveau MIN
Timer_Foerderband	TP	Timer (1min) für das Förderband
Foerderband_EIN	BOOL	Förderband zur Feststoffbeigabe
F_TRIG_0	F_TRIG	FB zur Erkennung einer negativen Flanke des Förderbandtimers
Heizung_EIN	BOOL	Heizung des Reaktors
Zeitanzeige	TIME	Anzeige der verstrichenen Zeit der Feststoffförderung

Tabelle 6: TAG-Liste Reaktorsteuerung

6.2.2.5. Programmcode

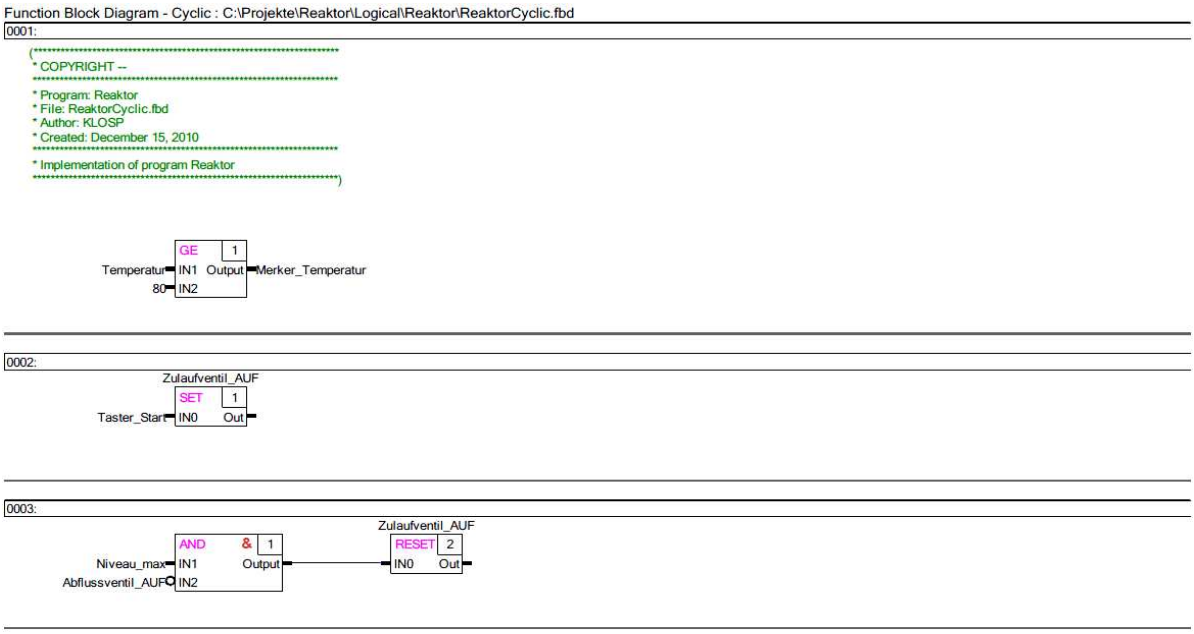

Abbildung 26: Programmcode Reaktorsteuerung - Teil 1

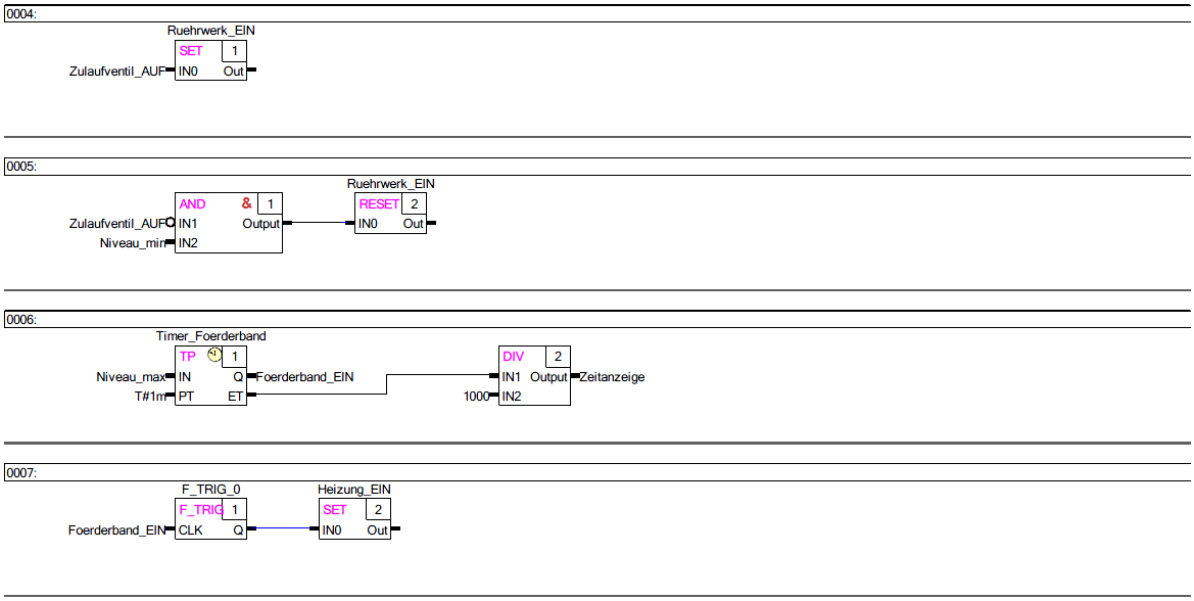

Abbildung 27: Programmcode Reaktorsteuerung - Teil 2

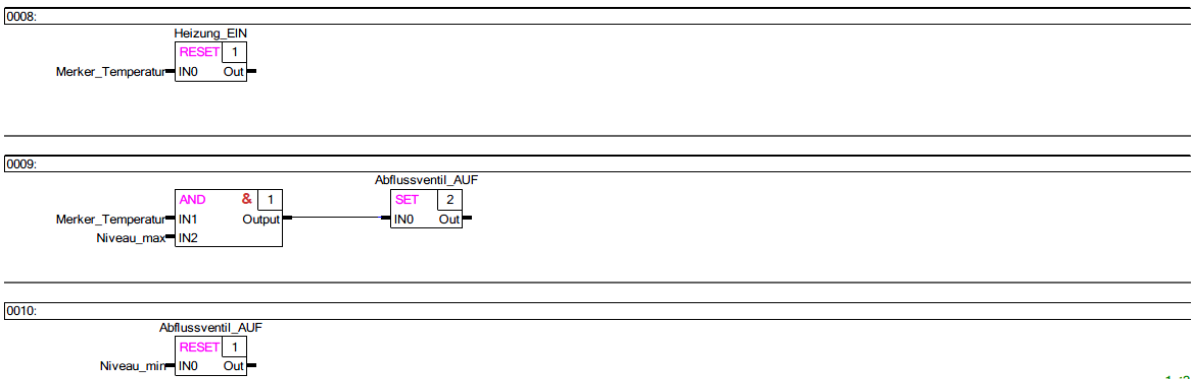

Abbildung 28: Programmcode Reaktorsteuerung - Teil 3

6.2.2.6. Visualisierung

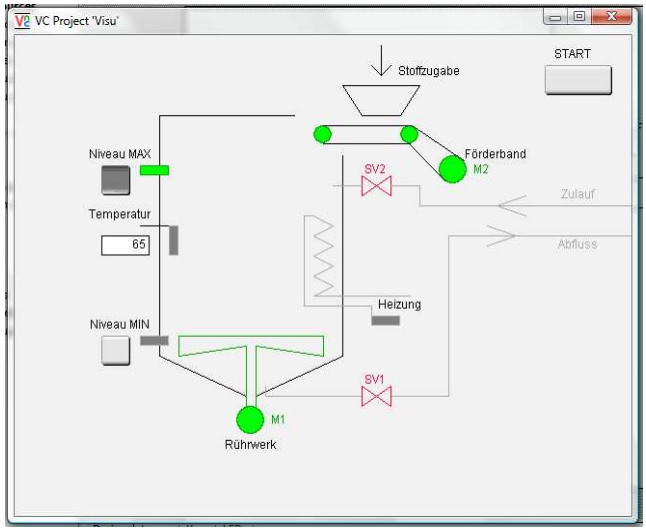

Abbildung 29: Visualisierung Reaktorsteuerung

6.2.3. Zusatz zu Reaktorsteuerung

6.2.3.1. Programmiersprache

Dieses Zusatzprogramm zum Beispiel „Reaktorsteuerung“ wurde mittels der Programmiersprache Automation Basic von B+R erstellt.

6.2.3.2. Funktionsbeschreibung

Mittels des Zusatzprogramms für das vorherige Beispiel „Reaktorsteuerung“ soll die Umschaltung zwischen 3 verschiedenen Rezepten ermöglicht werden. Es sollen folgende 3 verschiedene Rezepte eingebaut werden:

- Sollwert Temperatur 50℃, Sollwert Förderbandlaufz eit 30 Sek.
- Sollwert Temperatur 70℃, Sollwert Förderbandlaufz eit 60 Sek.
- Sollwert Temperatur 90℃, Sollwert Förderbandlaufz eit 120 Sek.

Die Rezepte sollen auf der Visualisierung auswählbar sein. Außerdem sollen die aktuell eingestellten Rezeptwerte auf der Visualisierung angezeigt werden.

6.2.3.3. Allgemeines

Das Programm baut auf dem Beispiel „Reaktorsteuerung“ auf, deshalb wird im Folgenden nur jener Teil des Automatisierungsprojektes beschrieben, der zusätzlich zum bereits vorhandenen Grundprogramm erstellt wurde. Die 3 Schalter zum Anwählen des Rezeptes wurden als Buttons auf dem virtuellen Panel realisiert. Die 2 Analoganzeigen wurden ebenfalls auf der Visualisierung integriert. Da hier nur ein bestehendes Programm ergänzt wird, wird der Programmcode als zusätzliches Programm im bestehenden Projekt eingebunden. In der nebenstehenden Grafik ist die Einbindung des zusätzlichen Programmcodes ersichtlich.

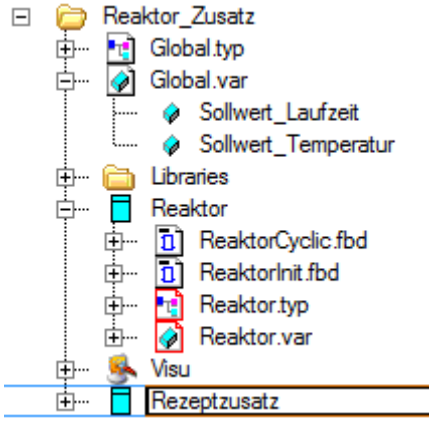

Abbildung 30: Einbindung Rezeptzusatz

6.2.3.4. TAG-Liste

Da es teilweise notwendig ist, dass beide Unterprogramme, also „Reaktor“ und „Rezeptzusatz“, auf die Variablen zugreifen müssen, werden in diesem Programm die Variablen in globale und lokale Variablen unterteilt.

Globale Variablen:

Project: Reaktor_Zusatz Global.var

Variable Declaration : C:\Projekte\Reaktor_Zusatz\Logical\Global.var

Name	Type	Value	Description [1]
********** * COPYRIGHT -- ********** * File: Global.var * Author: KLOSP * Created: December 15, 2010 ********** * Global variables of project Reaktor **********			
Sollwert_Laufzeit	TIME	T#0ms	Sollwert für die Laufzeit des Transportbandes
Sollwert_Temperatur	INT	0	Sollwert für die Reaktortemperatur

Tabelle 7: Globale TAG-Liste Zusatz Reaktorsteuerung

Lokale Variablen:

Project: Reaktor_Zusatz Rezeptzusatz.var

Variable Declaration : C:\Projekte\Reaktor_Zusatz\Logical\Rezeptzusatz\Rezeptzusatz.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Rezeptzusatz * File: Rezeptzusatz.var * Author: KLOSP * Created: January 04, 2011 ********** * Local variables of program Rezeptzusatz **********		
Rezept_1	BOOL	Rezept 1 aktiv
Rezept_2	BOOL	Rezept 2 aktiv
Rezept_3	BOOL	Rezept 3 aktiv

Tabelle 8: Lokale TAG-Liste Zusatz Reaktorsteuerung

6.2.3.5. Programmcode

Mittels Einbinden dieses Programmcodes wurde im Beispiel „Reaktorsteuerung“ die Umschaltung der Rezeptwerte realisiert.

```
Automation Basic : C:\Projekte\Reaktor_Zusatz\Logical\Rezeptzusatz\RezeptzusatzCyclic.ab

(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Rezeptzusatz
 * File: RezeptzusatzCyclic.ab
 * Author: KLOSP
 * Created: January 04, 2011
 *********************************************************************
 * Implementation of program Rezeptzusatz
 *********************************************************************)

PROGRAM _CYCLIC

if Rezept_1=true and not Rezept_2=true and not Rezept_3=true then
    Sollwert_Temperatur:=50
endif

if Rezept_1=true and not Rezept_2=true and not Rezept_3=true then
    Sollwert_Laufzeit:=T#30s
endif

if Rezept_2=true and not Rezept_1=true and not Rezept_3=true then
    Sollwert_Temperatur:=70
endif

if Rezept_2=true and not Rezept_1=true and not Rezept_3=true then
    Sollwert_Laufzeit:=T#1m
endif

if Rezept_3=true and not Rezept_1=true and not Rezept_2=true then
    Sollwert_Temperatur:=90
endif

if Rezept_3=true and not Rezept_1=true and not Rezept_2=true then
    Sollwert_Laufzeit:=T#2m
endif

END_PROGRAM
```

Abbildung 31: Programmcode Reaktorsteuerung

6.2.3.6. Visualisierung

Das Bild zeigt die in der Visualisierung ergänzten Buttons und Anzeigen zur Umsetzung der Rezeptumschaltung.

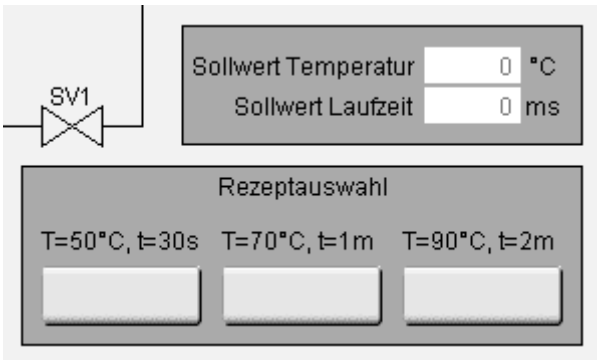

Abbildung 32: Visualisierung Reaktorsteuerung

6.2.4. Treibhaussteuerung

6.2.4.1. Programmiersprache

Dieses Programmbeispiel wurde mittels strukturiertem Text erstellt.

6.2.4.2. Funktionsbeschreibung

In einem Treibhaus soll durch Öffnen und Schließen der Dachfenster eine Regelung der Innentemperatur ermöglicht werden. Zur Ermittlung der aktuellen Temperatur wird ein Temperatursensor eingesetzt. Überschreitet diese Messung bestimmte Werte, so wird mit dem Öffnen der Fenster reagiert. Dabei wird nach folgenden Werten vorgegangen:

- Temperatur >=30℃ → Fenster 1 AUF
- Temperatur >=40℃ → Fenster 1 und Fenster 2 AUF
- Temperatur >=50℃ → Fenster 1 und Fenster 2 und Fenster 3 AUF

Das Schließen der Fenster erfolgt jeweils bei einer 5℃ tieferen Temperatur. Dadurch ergeben sich folgende Werte für das Schließen der Fenster:

- Temperatur <=25℃ → Fenster 3 und Fenster 2 und Fenster 1 ZU
- Temperatur <=35℃ → Fenster 3 und Fenster 2 ZU
- Temperatur <=45℃ → Fenster 3 ZU

Die Anlage soll über 2 Taster ein-/ausgeschaltet werden können. Ist die Anlage nicht aktiv, sollen alle Fenster geschlossen werden. Außerdem soll die Anlage mit einem Dämmerungsschalter ausgestattet sein, welcher ein Öffnen der Dachfenster in der Nacht verhindern soll.

6.2.4.3. Allgemeines

Die Temperaturmessung wurde als Eingabefeld am Panel umgesetzt. Die Ein-/Aus-Taster, die Endschalter für die Fensterstellungen und der Dämmerungsschalter sind als Buttons auf dem virtuellen Panel realisiert worden. Alle digitalen Ausgänge, wie zum Beispiel die Motoren der Fenster, wurden in die Visualisierung integriert und animiert.

6.2.4.4. TAG-Liste

Project: Treibhaus Treibhaussteuerung.var

Variable Declaration : C:\Projekte\Treibhaus\Logical\Treibhaussteuerung\Treibhaussteuerung.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Treibhaussteuerung * File: Treibhaussteuerung.var * Author: KLOSP * Created: January 08, 2011 ********** * Local variables of program Treibhaussteuerung **********		
Anlage_EIN	BOOL	Taster zum Einschalten der Anlage
Anlage_AKTIV	BOOL	Merker, dass Anlage derzeit aktiv ist
Temperatur	INT	Innentemperatur des Treibhauses
Daemmerungsschalter	BOOL	Dämmerungsschalter zum Vermeiden von offenen Fenstern in der Nacht
Fenster_1_AUF	BOOL	AUF Ansteuerung des Motors zu Fenster 1
Fenster_1_OFFEN	BOOL	Endschalter OFFEN des Fensters 1
Fenster_2_AUF	BOOL	AUF Ansteuerung des Motors zu Fenster 2
Fenster_2_OFFEN	BOOL	Endschalter OFFEN des Fensters 2
Fenster_3_AUF	BOOL	AUF Ansteuerung des Motors zu Fenster 3
Fenster_3_OFFEN	BOOL	Endschalter OFFEN des Fensters 3
Fenster_1_ZU	BOOL	ZU Ansteuerung des Motors zu Fenster 1
Fenster_1_GESCHLOSSEN	BOOL	Endschalter GESCHLOSSEN des Fensters 1
Fenster_2_ZU	BOOL	ZU Ansteuerung des Motors zu Fenster 2
Fenster_2_GESCHLOSSEN	BOOL	Endschalter GESCHLOSSEN des Fensters 2
Fenster_3_ZU	BOOL	ZU Ansteuerung des Motors zu Fenster 3
Fenster_3_GESCHLOSSEN	BOOL	Endschalter GESCHLOSSEN des Fensters 3
Anlage_AUS	BOOL	Taster zum Auschalten der Anlage

Tabelle 9: TAG-Liste Treibhaussteuerung

6.2.4.5. Programmcode

Structured Text : C:\Projekte\Treibhaus\Logical\Treibhaussteuerung\TreibhaussteuerungCyclic.st

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Treibhaussteuerung
 * File: TreibhaussteuerungCyclic.st
 * Author: KLOSP
 * Created: January 08, 2011
 *********************************************************************
 * Implementation of program Treibhaussteuerung
 *********************************************************************)

PROGRAM _CYCLIC


IF Anlage_EIN=TRUE THEN Anlage_AKTIV:=TRUE; END_IF;

IF Anlage_AUS=TRUE THEN Anlage_AKTIV:=FALSE; END_IF;

IF Anlage_AKTIV=TRUE AND Temperatur>=30 AND NOT Daemmerungsschalter=TRUE
    THEN Fenster_1_AUF:=TRUE; END_IF;
IF Fenster_1_OFFEN=TRUE OR Anlage_AKTIV=FALSE OR Daemmerungsschalter=TRUE
    THEN Fenster_1_AUF:=FALSE; END_IF;

IF Anlage_AKTIV=TRUE AND Temperatur>=40 AND NOT Daemmerungsschalter=TRUE
    THEN Fenster_2_AUF:=TRUE; END_IF;
IF Fenster_2_OFFEN=TRUE OR Anlage_AKTIV=FALSE OR Daemmerungsschalter=TRUE
    THEN Fenster_2_AUF:=FALSE; END_IF;

IF Anlage_AKTIV=TRUE AND Temperatur>=50 AND NOT Daemmerungsschalter=TRUE
    THEN Fenster_3_AUF:=TRUE; END_IF;
IF Fenster_3_OFFEN=TRUE OR Anlage_AKTIV=FALSE OR Daemmerungsschalter=TRUE
    THEN Fenster_3_AUF:=FALSE; END_IF;


IF Anlage_AKTIV=TRUE AND Temperatur<=25  OR Anlage_AKTIV=FALSE
    OR Daemmerungsschalter=TRUE THEN Fenster_1_ZU:=TRUE; END_IF;
IF Fenster_1_GESCHLOSSEN=TRUE THEN Fenster_1_ZU:=FALSE; END_IF;

IF Anlage_AKTIV=TRUE AND Temperatur<=35 OR Anlage_AKTIV=FALSE
    OR Daemmerungsschalter=TRUE THEN Fenster_2_ZU:=TRUE; END_IF;
IF Fenster_2_GESCHLOSSEN=TRUE THEN Fenster_2_ZU:=FALSE; END_IF;

IF Anlage_AKTIV=TRUE AND Temperatur<=45 OR Anlage_AKTIV=FALSE
    OR Daemmerungsschalter=TRUE THEN Fenster_3_ZU:=TRUE; END_IF;
IF Fenster_3_GESCHLOSSEN=TRUE THEN Fenster_3_ZU:=FALSE; END_IF;


END_PROGRAM
```

Abbildung 33: Programmcode Treibhaussteuerung

6.2.4.6. Visualisierung

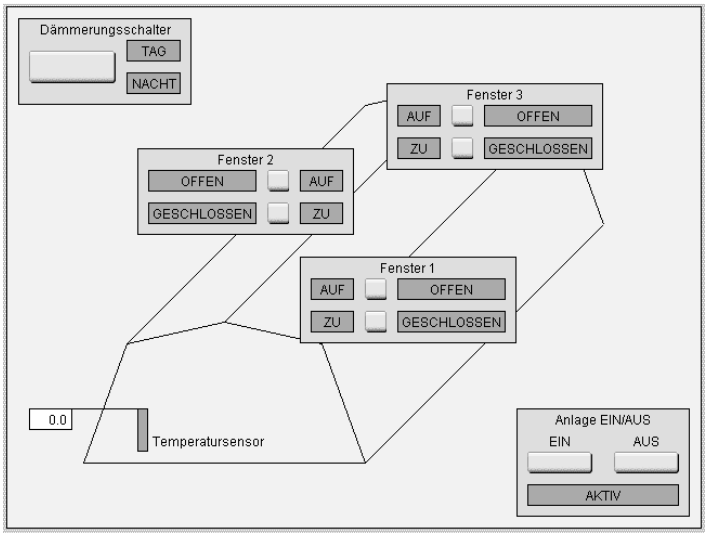

Abbildung 34: Visualisierung Treibhaussteuerung

6.2.5. Tunnelventilationssteuerung

6.2.5.1. Programmiersprache

Dieses Beispiel wurde in „ANSI C“ programmiert.

6.2.5.2. Funktionsbeschreibung

Die Belüftung eines Tunnels erfolgt durch 3 separat angesteuerte Ventilatoren. Wie viele Ventilatoren benötigt werden wird über 3 Abgassensoren festgestellt. Dabei wird nach folgenden Vorgaben gearbeitet:

- 1 Sensor aktiv → 1 Ventilator aktiv
- 2 Sensoren aktiv → 2 Ventilatoren aktiv
- 3 Sensoren aktiv → 3 Ventilatoren aktiv

Der erste Ventilator soll ohne Verzögerung einschalten, die beiden folgenden Ventilatoren sollen mit einer Einschaltverzögerung von 5 Sekunden aktiviert werden.

6.2.5.3. Allgemeines

Die Sensoren wurden als Buttons auf dem virtuellen Panel realisiert, die Ventilatoren wurden in die Visualisierung integriert und animiert.

6.2.5.4. TAG-Liste

Project: Tunnelbelueftung Tunnelsteuerung.var

Variable Declaration : C:\Projekte\Tunnelbelueftung\Logical\Tunnelsteuerung\Tunnelsteuerung.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Tunnelsteuerung * File: Tunnelsteuerung.var * Author: KLOSP * Created: January 08, 2011 ********** * Local variables of program Tunnelsteuerung **********		
Sensor_2	BOOL	Abgassensor 2
Sensor_1	BOOL	Abgassensor 1
Sensor_3	BOOL	Abgassensor 3
Ventilator_3	BOOL	Ventilator 3
Ventilator_2	BOOL	Ventilator 2
Ventilator_1	BOOL	Ventilator 1
TON_0	TON	Timer zur Einschaltverzögerung des 2. Ventilators
TON_1	TON	Timer zur Einschaltverzögerung des 3. Ventilators

Tabelle 10: TAG-Liste Tunnelventilationssteuerung

6.2.5.5. Programmcode

ANSI C : C:\Projekte\Tunnelbelueftung\Logical\Tunnelsteuerung\TunnelsteuerungCyclic.c

```
/********************************************************************
 * COPYRIGHT --
 ********************************************************************
 * Program: Tunnelsteuerung
 * File: TunnelsteuerungCyclic.c
 * Author: KLOSP
 * Created: January 08, 2011
 ********************************************************************
 * Implementation of program Tunnelsteuerung
 ********************************************************************/

#include <bur/plctypes.h>

#ifdef _DEFAULT_INCLUDES
 #include <AsDefault.h>
#endif

void _CYCLIC TunnelsteuerungCyclic( void )

{

TON_0.IN = (Sensor_1 && Sensor_2) || (Sensor_1 && Sensor_3) || (Sensor_3 && Sensor_2);
TON_0.PT = 5000;
TON(&TON_0);

TON_1.IN = Sensor_1 && Sensor_2 && Sensor_3;
TON_1.PT = 5000;
TON(&TON_1);

Ventilator_3 = (Sensor_1 && Sensor_2 && Sensor_3) && TON_1.Q;

Ventilator_2 = ((Sensor_1 && Sensor_2) || (Sensor_1 && Sensor_3) || (Sensor_3 && Sensor_2)) && TON_0.Q;

Ventilator_1 = Sensor_1  || Sensor_2 || Sensor_3;

}
```

Abbildung 35: Programmcode Tunnelbelüftung

6.2.5.6. Visualisierung

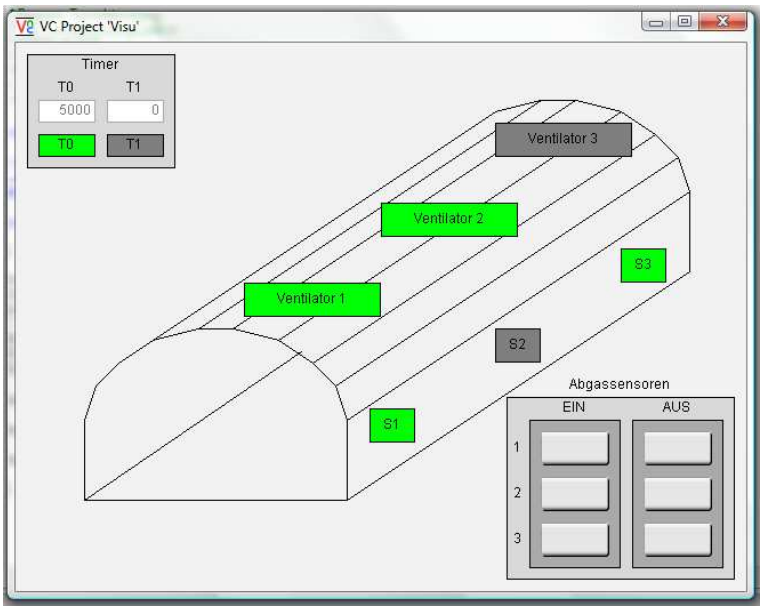

Abbildung 36: Visualisierung Tunnelventilationssteuerung

6.2.6. Wasserheizung mit Sonnenkollektor

6.2.6.1. Programmiersprache

Dieses Programmbeispiel wurde in Form einer AWL programmiert.

6.2.6.2. Funktionsbeschreibung

Ein Heizkessel soll mittels eines Sonnenkollektors aufgeheizt werden. Dazu soll, solange die Temperatur im Sonnenkollektor größer ist als jene im Heizkessel, die Umwälzpumpe laufen.

Die Entnahme des Warmwassers soll nur möglich sein, wenn die Temperatur im Heizkessel höher als 35℃ ist. Während die Warmwasserentnahme über das W armwasserventil läuft, soll über das Kaltwasserventil kaltes Wasser nachgefördert werden, damit das Niveau im Heizkessel gehalten wird.

6.2.6.3. Allgemeines

Die Temperaturmessungen des Sonnenkollektors und des Heizkessels wurden in diesem Beispielprogramm durch Eingabefelder auf dem virtuellen Panel ersetzt. Auch der Taster zum Starten der Warmwasserentnahme wurde lediglich als Button auf dem Panel erstellt und umgesetzt. Die im Beispiel enthaltenen Aktoren (Freigabeanzeige, Anzeige der Kesseltemperatur, Umwälzpumpe, Warmwasserventil und Kaltwasserventil) wurden symbolisch auf dem Panel visualisiert und animiert.

6.2.6.4. TAG-Liste

Project: Wasserheizung_Sonnenkollektor Wasserheizung.var

Variable Declaration : C:\Projekte\Wasserheizung_Sonnenkollektor\Logical\Wasserheizung\Wasserheizung.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Wasserheizung * File: Wasserheizung.var * Author: KLOTZ MARTIN * Created: December 18, 2010 ********** * Local variables of program Wasserheizung **********		
Kollektortemperatur	INT	Temperaturmessung im Sonnenkollektor
Kesseltemperatur	INT	Temperaturmessung im Heizkessel
Pumpe_EIN	BOOL	Umwälzpumpe für Heizkreislauf
Taster_WW_Entnahme	BOOL	Taster zum Starten der Warmwasserentnahme
Warmwasserventil	BOOL	Ventil zur Warmwasserentnahme
Freigabe_WW	BOOL	Ventil zur Kaltwasserzufuhr

Tabelle 11: TAG-Liste Wasserheizung

6.2.6.5. Programmcode

```
PROGRAM _CYCLIC

  LD      Kollektortemperatur     // Lade Kollektortemperatur
  SUB     5                       // Kollektortemperatur - 5°C
  GE      Kesseltemperatur        // >=Kesseltemperatur ?
  S     Pumpe_EIN                 // wenn ja, setze Umwälzpumpe EIN

  LD      Kesseltemperatur        // Lade Kesseltemperatur
  GE      35                      // >= 35 °C ?
  S     Freigabe_WW               // wenn ja, setze Freigabe Warmwasserentnahme EIN

  LD      Taster_WW_Entnahme      // Lade Start-Taster
  AND     Freigabe_WW             // Freigabe Warmwasserentnahme EIN
  S     Warmwasserventil          // wenn beides aktiv, Warmwasserventil öffnen

  LD      Kollektortemperatur     // Lade Kollektortemperatur
  SUB     5,01                    // - 5,01 °C
  LE      Kesseltemperatur        // <=Kesseltemperatur
  R     Pumpe_EIN                 // wenn ja, Umwälzpumpe rücksetzen

  LD      Kesseltemperatur        // Lade Kesseltemperatur
  LE      34,99                   // <= 34,99 °C
  R     Warmwasserventil          // wenn ja, Warmwasserventil rücksetzen
  R     Freigabe_WW               // wenn ja, Freigabe Warmwasserentnahme rücksetzen



END_PROGRAM
```

Abbildung 37: Programmcode Wasserheizung

6.2.6.6. Visualisierung

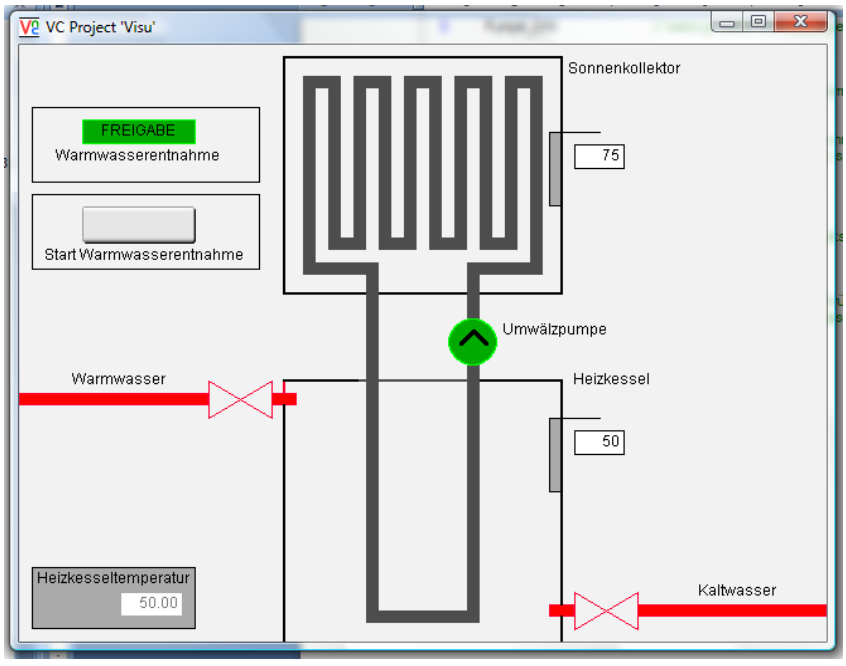

Abbildung 38: Visualisierung Wasserheizung

6.2.7. Beispiel Feldgeräteanbindung

6.2.7.1. Programmiersprache

Das Beispiel wurde als KOP (LD) ausgeführt.

6.2.7.2. Funktionsbeschreibung

Mit Hilfe dieses Programmes ist es möglich, verschiedene Feldgeräte über ein virtuelles Bedienpanel zu steuern bzw. zu beobachten.

Funktion 1: Eine Temperaturmessung (PT100 mit Kopftransmitter) wird an einen Analogeingang angeschlossen und als Temperaturanzeige in die Visualisierung integriert.

Funktion 2: Ein I/P–Wandler (Konverter von Strom in pneumatischen Druck) wird an einen analogen Ausgang angeschlossen und soll 0…20 mA in einen pneumatischen Druck von 0,2…1 bar umwandeln.

Funktion 3: Durch 2 Buttons auf dem virtuellen Panel wird ein Magnetventil über einen Digitalausgang angesteuert. Über dieses Magnetventil wird ein pneumatischer Antrieb mit einem Druck von 6 bar beaufschlagt, wodurch dieser geschalten wird. Mittels einer Endschalterbox am Pneumatik-Antrieb wird eine Rückmeldung über die aktuelle Stellung des Antriebs an einen Digitaleingang des Steuerungssystems ausgegeben, welche auf der Visualisierung dargestellt wird.

6.2.7.3. Allgemeines

Dieses Beispiel soll die Anbindung diverser Feldgeräte veranschaulichen. Dazu wurden folgende Komponenten an die I/O-Baugruppen des Testsystems angeschlossen und im Programm verarbeitet:

- Temperatursensor (PT100) inkl. Kopftransmitter für 4…20 mA bei einem Messbereich von 0 bis +50℃
- I / P – Wandler (0…20 mA auf 0,2…1 bar)
- Magnetventil zur pneumatischen Steuerung eines pneumatischen Antriebs
- Endschalterbox für Rückmeldung der Stellung des pneumatischen Antriebs

Durch diese Feldgeräte soll die Anbindung mittels digitaler Ein-/Ausgänge und analoger Ein-/Ausgänge aufgezeigt werden.

6.2.7.4. TAG-Listen

TAG-Liste zu Programm „Temperaturmessung“:

Project: COM Temperaturmessung.var

Variable Declaration : C:\Projekte\COM\Logical\Temperaturmessung\Temperaturmessung.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: NewProgram * File: NewProgram.var * Author: KLOSP * Created: December 23, 2010 ********** * Local variables of program NewProgram **********		
Ana_In_1	INT	Analoger Eingangswert der Temperaturmessung
Temperatur_Faktor	REAL	Faktor Temperaturmessung (Ana_In_1 / Anzahl der Inkremente)
Temperatur_skaliert	REAL	Temperaturwert skaliert auf 0 bis 50°C
Temperatur_unskaliert	REAL	Unskalierter Temperaturwert (in Inkrementen)

Tabelle 12: TAG-Liste Feldgeräteanbindung/Temperaturmessung

TAG-Liste zu Programm „IP Wandler“:

Project: COM IP_Wandler.var

Variable Declaration : C:\Projekte\COM\Logical\IPWandler\IP_Wandler.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: IP_Wandler * File: IP_Wandler.var * Author: KLOSP * Created: December 27, 2010 ********** * Local variables of program IP_Wandler **********		
Eingabewert_Druck	REAL	Wert des gewünschten Druckes aus Eingabefeld der Visualisierung
Sollwert_Druck	REAL	Eingabewert - Offset
Analogausgang_1	REAL	Analogausgangswert REAL
Analogausgang_1_INT	INT	Analogausgangswert INT
Hilfswert_1	REAL	Hilfswert zur Anzeige des mA-Signals zu Ausgang 1
Sollwert_Analogausgang	REAL	Sollwert_Druck / Messbereich
mA_Wert	REAL	mA-Wert für Anzeige auf Visualisierung

Tabelle 13: TAG-Liste Feldgeräteanbindung/IP Wandler

TAG-Liste zu Programm „Ventilsteuerung“:

Project: COM Ventilsteuerung.var

Variable Declaration : C:\Projekte\COM\Logical\Ventilsteuerung\Ventilsteuerung.var

Name	Type	Description [1]
********** * COPYRIGHT -- ********** * Program: Ventilsteuerung * File: Ventilsteuerung.var * Author: KLOSP * Created: December 27, 2010 ********** * Local variables of program Ventilsteuerung **********		
Ventil_AUF	BOOL	Taster, um Ventil AUF zu schalten
Ansteuerung_AUF	BOOL	Ventil-Ansteuerung für Stellung AUF
Ventil_ZU	BOOL	Taster, um Ventil ZU zu schalten
Rueckmeldung_OFFEN	BOOL	Rückmeldung, dass das Ventil offen ist
Ventil_OFFEN	BOOL	BIT für OFFEN-Anzeige des Ventils
Ventil_GESCHLOSSEN	BOOL	BIT für OFFEN-Anzeige des Ventils

Tabelle 14: TAG-Liste Feldgeräteanbindung/Ventilsteuerung

6.2.7.5. Programmcodes

Programmcode zu „Temperaturmessung“:

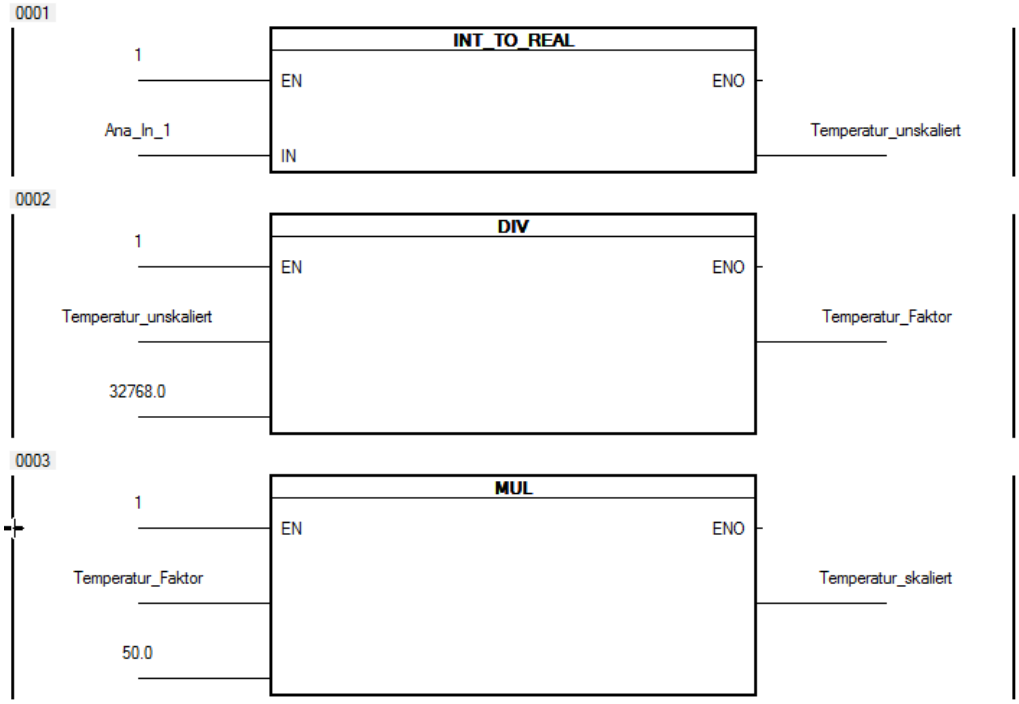

Abbildung 39: TAG-Liste Feldgeräteanbindung/Temperaturmessung

Programmcode zu „IP Wandler“:

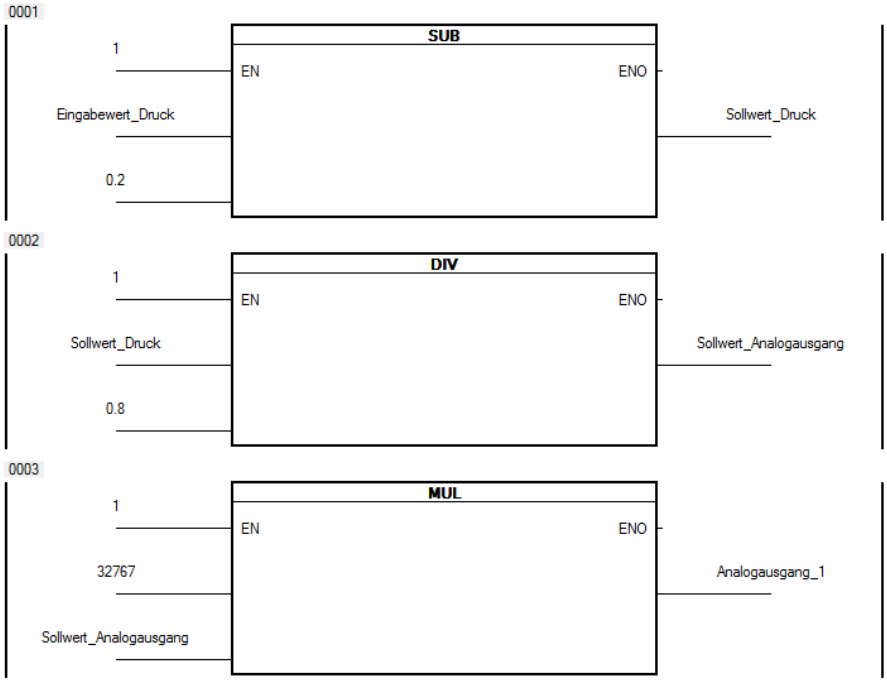

Abbildung 40: TAG-Liste Feldgeräteanbindung/IP Wandler – Teil 1

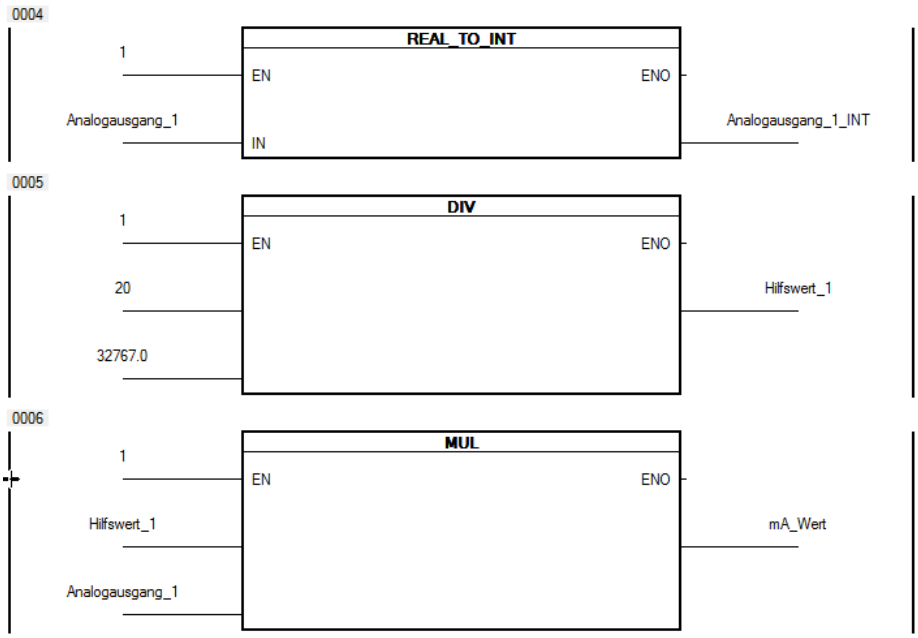

Abbildung 41: TAG-Liste Feldgeräteanbindung/IP Wandler – Teil 2

Programmcode zu „Ventilsteuerung“:

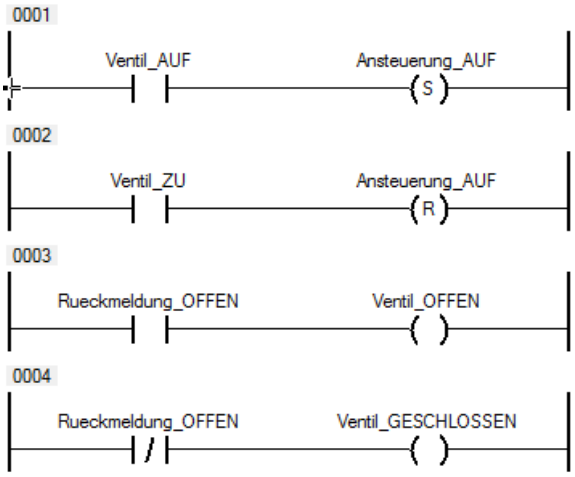

Abbildung 42: TAG-Liste Feldgeräteanbindung/Ventilsteuerung

6.2.7.6. I/O Mapping

Um die im Projekt angelegten Variablen (Tags) mit den entsprechenden Ein-/Ausgängen zu koppeln, muss das so genannte I/O-Mapping durchgeführt werden. Darunter versteht man das Zuweisen der Prozessvariablen im Programm zu den, in der HW-Konfiguration enthaltenen, I/O-Baugruppen. In unten ersichtlichem Beispiel wird die Prozessvariable „Ana_In_1“ des Programms „Temperatur“ an „AnalogInput01“ gekoppelt und entspricht somit dauernd dem Wert der an diesem Anschluss der Eingangskarte anliegt.

AnalogInput01	INT	Automatic	Temperatur.Ana_In_1			\IoMap.iom	±10 V / 0 to 20 mA, resolution 12 bit

Tabelle 15: Beispiel I/O Mapping

Über die Konfiguration der I/O-Baugruppen kann der Messbereich des Eingangs geändert werden. Selbiges wurde in diesem Beispiel ebenfalls für den Analogausgang zur Steuerung des IP Wandlers und für den digitalen Ein- und Ausgang zur Steuerung des pneumatischen Ventilantriebes durchgeführt.

6.2.7.7. Visualisierung

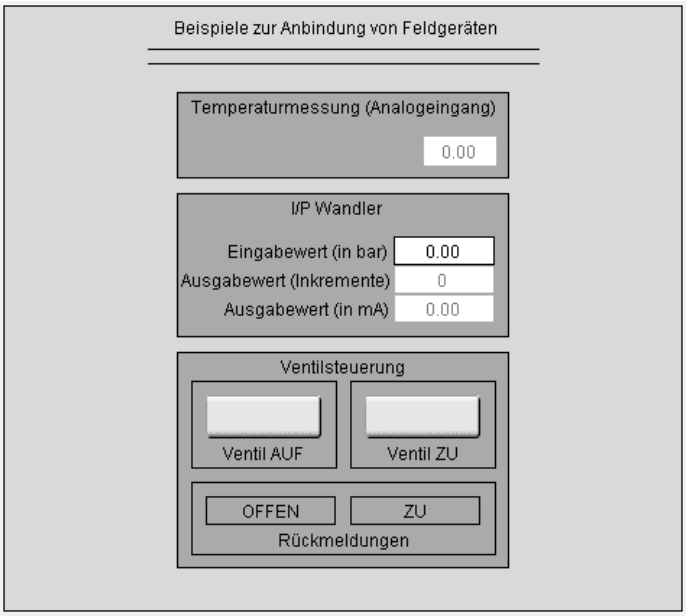

Abbildung 43: Visualisierung Feldgeräteanbindung

6.2.7.8. Bilder der Versuchsaufbauten

Im Folgenden sind die Versuchsaufbauten abgebildet, welche im Zuge dieser Diplomarbeit erstellt wurden.

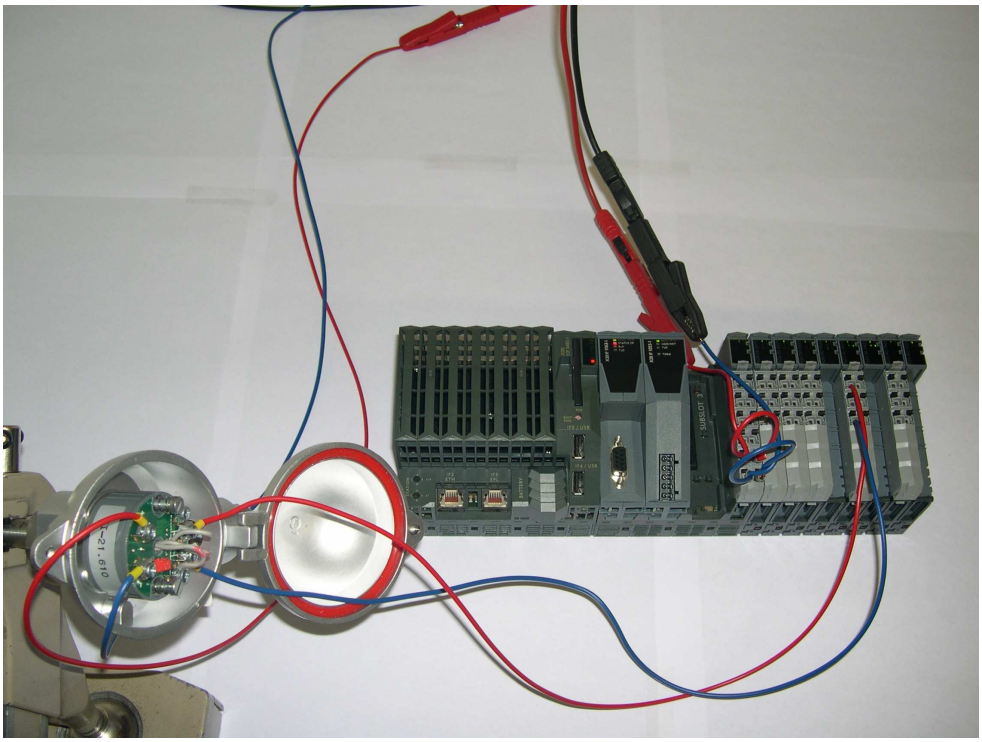

Abbildung 44: Anlagen - Anbindung einer Temperaturmessung

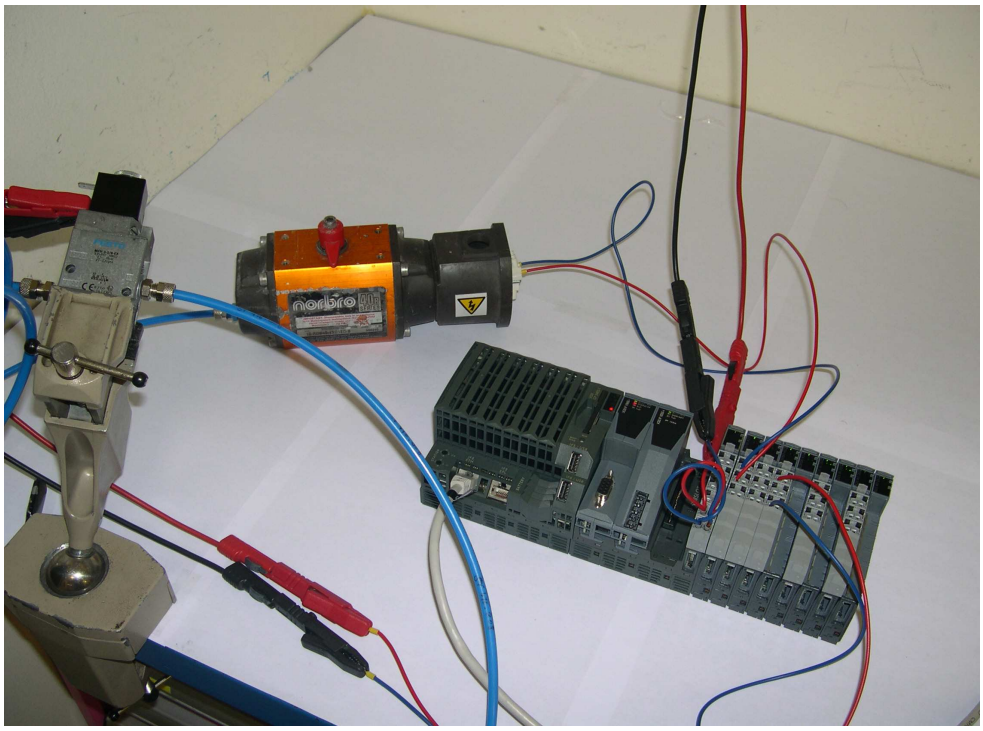

Abbildung 45: Anlagen - Steuerung eines pneumatischen Antriebs mittels Magnetventil

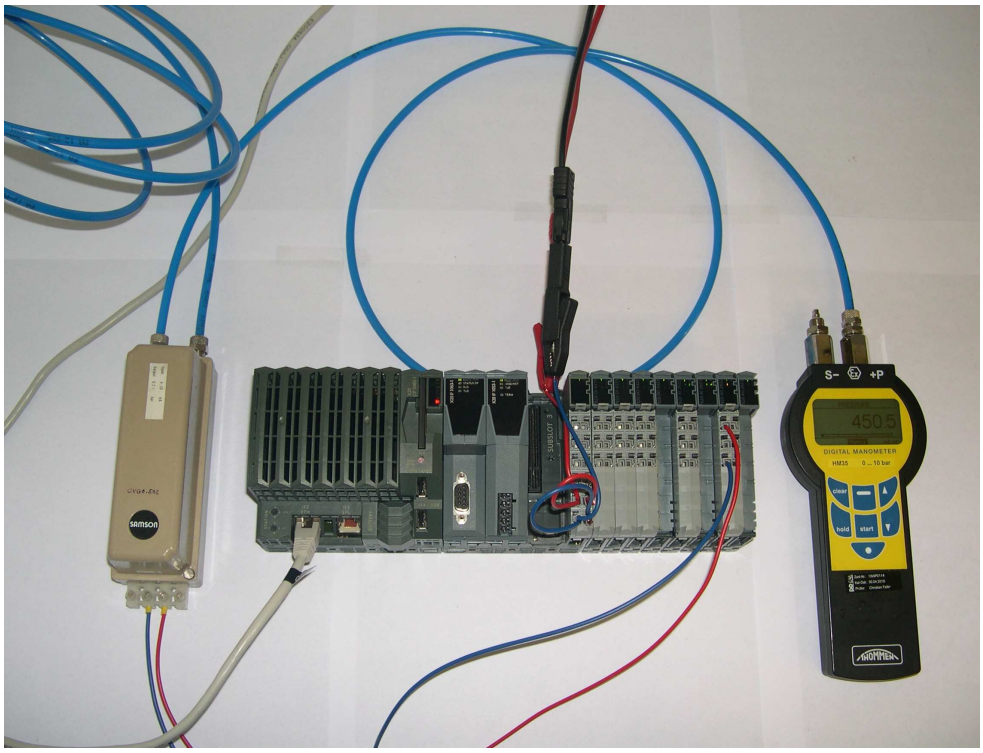

Abbildung 46: Anlagen - Anbindung I/P Wandler

7. Konfiguration von Kommunikationsverbindungen

7.1. Feldbusse[53]

In Automation Studio ist es möglich, Module zur Anbindung an Feldbusse, wie ein I(O-Modul in der Physical View einzufügen und im Anschluss daran zu konfigurieren. Durch die Importfunktionen der digitalen Gerätebeschreibungen der jeweiligen Anbieter ist stehen dem User damit einheitliche Schnittstellen zur Verfügung.

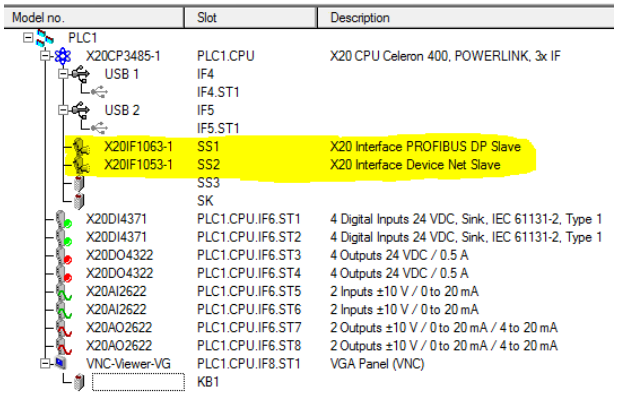

Abbildung 47: Einfügen der Kommunikationsmodule

Folgende Punkte sind zur Konfiguration einer Feldbuskommunikationsanbindung notwendig:

- Module in den Hardware-Baum (Physical View) einfügen
- I/O-Konfiguration durchführen, Master u. Slave-Einstellungen vornehmen
- Zuweisung von Prozessvariablen an die I/O's
- Falls notwendig, Bibliotheken für Spezialzugriffe nutzen
- Anschließen der I/O Kanäle des Feldbusses

7.2. OPC „OLE for Process Control"[54]

Bei OPC handelt es sich um einen Industriestandard, der von den führenden Firmen der Automatisierungsbranche in Kooperation mit Microsoft entwickelt wurde. Verwaltet und betreut wird dieser Standard von der OPC Foundation.

Bei den Systemen der Firma B+R kann der Zugriff eines OPC Clients entweder auf einen Server in Form eines Windows-PC's oder oder einen Server, der in ein Echtzeit-Betriebssystem integriert ist. Die Schnittstelle ist standardisiert und kann dadurch vom Anwender durch Auswahl von SCADA Paketen (Supervisory Control and Data Acquisition) genutzt werden. Es besteht jedoch auch die Möglichkeit einen selbst erstellten OPC Client, basierend auf Visual Studio.NET, VC++ oder VB, zu nutzen.

Automation Studio bietet dem User also die Möglichkeit, durch den direkten Zugriff auf die Prozessvariablen die OPC Konfiguration selbst zu erstelle und zu verwalten.

53 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm [05.01.2011]
54 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm [05.01.2011]

7.3. PVI Kommunikation „B+R ←→ Windows“[55]

Das Process Visualization Interface (PVI) als Zugang zum B+R Automation Net bietet allen auf Windows basierenden Softwarepaketen eine gemeinsame Schnittstelle zum Automatisierungsnetzwerk. Dadurch bieten sich folgende Möglichkeiten:

- Kommunikationsdienste für Automation Studio
- DLL Schnittstelle zum Erstellen von Visual Basic, VC++, Delphi Anwendungen und Treibern
- PVI-Services DLL zur Anbindung von .NET Anwendungen an B+R Steuerungssysteme
- Offenheit für andere Architekturen, wie zum Beispiel PVI-OPC, PVI-DDE, PVI-Web, ...
- Unterstützung von PC- und Steuerungsphysik (RS232, Ethernet, Modem, PCI Bus)

55 URL http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm

Schlusswort

Die angebotene Hardware des X20 Systems ist bestens ausgestattet und bietet für nahezu alle Anwendungen in der Automatisierungstechnik die entsprechenden Komponenten. Die Einsatzgebiete für B+R Steuerungssysteme reichen also von einfachsten Steuerungsanwendungen bis hin zur Automatisierung von großen Industrieanlagen und komplexen Prozessabläufen. Die Hardware ist mechanisch kompakt ausgeführt und dadurch ideal für den Einbau in Schaltschränken konzipiert.

In Sachen Industrietauglichkeit bietet B+R noch einige sehr komfortable Funktionen an. Es lassen sich beispielsweise alle I/O Baugruppen während des Betriebes austauschen, ohne Einfluss auf das gerade laufende Automatisierungsprojekt der Zentraleinheit zu haben. Auch Softwareänderungen sind problemlos ohne Unterbrechung des Betriebes einbindbar. Besonders hervorzuheben ist die Tauglichkeit der B+R Komponenten für den Serienmaschinenbau. Durch die Verwendung der CF (Compact Flash) Speicherkarten auf den CPU's kann die gesamte Hardwarekonfiguration und das Programm inkl. Prozessvariablen und Symbolik ohne eine vorhandene Zentraleinheit bereits auf der Speicherkarte abgelegt werden. Dadurch ergibt sich die Möglichkeit, auch Maschinen mit verschiedenen Konfigurationen bereits vollständig zusammenzustellen und im Anschluss durch Stecken der entsprechend vorbereiteten CF-Karte in Betrieb zu nehmen.

Die Software zur Konfiguration und Programmierung der B+R Systeme, Automation Studio, stellte sich während dieser Arbeit als besonders leistungsfähig heraus. Ein wesentlicher positiver Aspekt ist, dass es eine Software für alle Abschnitte eines Projektes gibt, was B+R von diversen anderen Steuerungsanbietern abhebt, bei denen mehrere Programme notwendig sind um ein vollständiges Automatisierungsprojekt zu erstellen. Auch hervorzuheben sind das laufzeitfähige Simulationsprogramm AR000, welches dem Benutzer in Zusammenarbeit mit einem VNC Player ermöglicht das gesamte Projekt inkl. Visualisierung direkt am Programmiergerät zu testen. Auch das integrierte Tool zur Erstellung von Visualisierungen ist sehr leistungsstark. Es ermöglicht die Erstellung von optisch durchaus ansprechenden Prozessvisualisierungen und Animationen in verhältnismäßig kurzer Zeit. In Bezug auf Kompatibilität mit diversen Programmiersprachen ist Automation Studio auch sehr gut aufgestellt. Bis auf die Ablaufsprache (SFC) sind alle Programmiersprachen der IEC 61131-3 mit den Systemen kompatibel. Zusätzlich zu den in der Norm enthaltenen Sprachen werden noch Editoren zur Programmierung mittels „Automation Basic", „Continous Function Chart" und „ANSI C" angeboten. Insgesamt bietet B+R also sehr breit gefächerte Möglichkeiten zur Programmerstellung und auch die Möglichkeit die verschiedenen Editoren in einem Automatisierungsprojekt miteinander zu kombinieren.

Ein weiterer Punkt dieser Arbeit war die Untersuchung des industriellen Ethernet-Standards „Ethernet Powerlink". Die Untersuchungen ergaben, dass es sich bei Ethernet Powerlink um ein enorm leistungsfähiges Netzwerk handelt, das in Punkten wie Verarbeitungsgeschwindigkeit, Wartung und Durchgängigkeit durch mehrere Ebenen von Komplettsystemen aufzeigt.

Danksagung

Abschließend möchte ich mich noch bei folgenden Personen und Firmen bedanken, durch deren Kooperation die Erstellung dieser Arbeit möglich gemacht wurde.

Zunächst möchte ich mich bei Herrn Prof. Dr. Ing. Dietmar Römer, welcher die Arbeit seitens der Hochschule Mittweida betreute, und bei Herrn Dipl. Ing. Hohenwarter Albert, der die Betreuung seitens der Firma SANDOZ GesmbH übernahm, bedanken.

Besonders bedanken darf ich mich auch bei der Firma Bernecker und Rainer Industrie Elektronik GesmbH in Eggelsberg. Diese stellte mir für die Dauer dieser Abschlussarbeit kostenlos diverse Steuerungskomponenten zur Verfügung, die zur Erstellung der Versuchsaufbauten notwendig waren. Besonderer Dank gilt hier Herrn Thomas Dicker, Frau Martha Weinberger und Frau Karin Nobis, welche mir die Leihgeräte zur Verfügung stellten und die gesamte Organisation seitens B+R übernahmen.

Holzgau, Jänner 2011

(Martin KLOTZ)

Literaturverzeichnis

1 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/company_DEU_HTML.htm> , 01.12.2010

2 Bernecker und Rainer GesmbH: Anwenderhandbuch System 2003: 2000

3 Bernecker und Rainer GesmbH: Anwenderhandbuch System 2003: 2000

4 Bernecker und Rainer GesmbH: Anwenderhandbuch System 2005: 2004

5 Bernecker und Rainer GesmbH: Anwenderhandbuch System 2003: 2000

6 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

7 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

8 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

9 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

10 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

11 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

12 3S Smart Software Solutions GmbH: Website 3S Software Solutions. URL: < http://www.3s-software.com/index.shtml?de_homepage>, 15.12.2010

13 3S Smart Software Solutions GmbH: Website 3S Software Solutions. URL: < http://www.3s-software.com/index.shtml?codesys_dev_dir >, 15.12.2010

14 Wagner, Roland: Information über E-Mail: Kompatibilität mit CoDeSys: r.wagner@3s-software.com, 17.12.2010

15 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

16 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

17 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

18 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

19 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

20 Bernecker und Rainer GesmbH: Broschüre zu Automation Studio: 2007

21 FH Düsseldorf: IEC 61131-Grundlagen. URL: <http://control-net.fh-duesseldorf.de/IEC61131/iec_d/iec11.htm>, 05.12.2010

22 John, Karl Heinz; Tiegelkamp, Michael: SPS-Programmierung mit IEC 61131-3: Konzepte und Programmiersprachen: 4., neubearbeitete Auflage: Springer-Verlag Berlin Heidelberg, 2009

23 Lepers, Heinrich: SPS Programmierung nach IEC 61131-3: Mit Beispielen für CoDeSys und STEP 7: 1. Auflage: Franzis Verlag GmbH Poing, 2005

24 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm> , 05.12.2010

25 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm> , 05.12.2010

26 John, Karl Heinz; Tiegelkamp, Michael: SPS-Programmierung mit IEC 61131-3: Konzepte und Programmiersprachen: 4., neubearbeitete Auflage: Springer-Verlag Berlin Heidelberg, 2009

27 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm> , 05.12.2010

28 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm> , 05.12.2010

29 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation. URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue /hs.xsl/products_151728_DEU_HTML.htm> , 05.12.2010

30 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

31 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

32 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

33 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

34 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

35 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

36 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

37 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

38 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

39 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

40 Bernecker und Rainer GesmbH: Automation Studio V3.0.81.18: Automation Studio Content Help, 2010

41 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation: Marktstudie zu Ethernet Powerlink: URL: < http://www.br-automation.com/cps/rde/xchg/br-automation_com/hs.xsl/cookies_allowed.htm?caller=news_14230_DEU_HTML.htm> , 29.11.2010

42 HMS Industrial Networks GmbH: Website HMS Industrial Networks: URL: <http://www.anybus.de/technologie/ethernet.shtml>, 03.12.2010

43 HMS Industrial Networks GmbH: Website Feldbusse.de: URL: < http://feldbusse.de/Profibus/profibus.shtml>, 03.12.2010

44 HMS Industrial Networks GmbH: Website Feldbusse.de: URL: < http://feldbusse.de/DeviceNet/devicenet.shtml >, 03.12.2010

45 Vector Informatik GmbH: Website e-learning Vector Informatik: URL: < http://www.vector-elearning.com/vl_einfuehrungcan_de.html>, 01.12.2010

46 Beckhoff Automation GmbH: Beckhoff Information System: URL: < http://infosys.beckhoff.com/index.php?content=../content/1031/fc510x/HTML/CO_comFieldbus.htm&id=>, 01.12.2010

47 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

48 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

49 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

50 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

51 Bernecker und Rainer GesmbH: Anwenderhandbuch System X20: 2009

52 CHIP Xonio Online GmbH: CHIP Online DE: URL:< http://www.chip.de/downloads/RealVNC_12997724.html>, 18.12.2010

53 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation: URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm> , 05.01.2011

54 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation: URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm> , 05.01.2011

55 Bernecker und Rainer GesmbH: Website Bernecker und Rainer Industrie Automation: URL: < http://www.br-automation.com/cps/rde/xchg/br-productcatalogue/hs.xsl/products_151765_DEU_HTML.htm> , 05.01.2011

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Holzgau, am 24. Jänner 2011

(Martin KLOTZ)